Detecting Personal Life Events from Social Media

Thesis

Detecting Personal Life Events from Social Media

A THESIS PRESENTED
BY
THOMAS K. DICKINSON
TO
THE DEPARTMENT OF SCIENCE, TECHNOLOGY, ENGINEERING AND MATHEMATICS

IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN THE SUBJECT OF
COMPUTER SCIENCE

THE OPEN UNIVERSITY
MILTON KEYNES, ENGLAND

MAY 2019

Thesis advisor: Professor Harith Alani & Dr Paul Mulholland Thomas K. Dickinson

Detecting Personal Life Events from Social Media

Abstract

Social media has become a dominating force over the past 15 years, with the rise of sites such as Facebook, Instagram, and Twitter. Some of us have been with these sites since the start, posting all about our personal lives and building up a digital identify of ourselves.

But within this myriad of posts, what actually matters to us, and what do our digital identities tell people about ourselves? One way that we can start to filter through this data, is to build classifiers that can identify posts about our personal life events, allowing us to start to self reflect on what we share online.

The advantages of this type of technology also have direct merits within marketing, allowing companies to target customers with better products. We also suggest that the techniques and methodologies built throughout this thesis also have opportunities to support research within other areas such as cyber bullying, and radicalisation detection.

The aim of this thesis is to build upon the under researched area of life event detection, specifically targeting Twitter, and Instagram. Our goal is to develop classifiers that identify a list of life events inspired by cognitive psychology, where we target a total of seven within this thesis.

To achieve this we look to answer three research questions covered in each of our empirical chapters. In our first empirical chapter, we ask; *What features would improve the classification of important life events.* To answer this, we look at first extracting a new dataset from Twitter targeting the following events: Getting Married, Having Children, Starting School, Falling in Love, and Death of a Parent. We look at three new feature sets: interactions, content, and semantic features, and compare against a current state of the art technique.

In our second empirical chapter, we draw inspiration from cheminformatics, and frequent sub-graph mining to ask; *Could the inclusion of semantic and syntactic patterns improve performance in our life event classifier.* Here we look at expanding our tweets into semantic networks, as well as consider two forms of syntactic relationships between tokens. We then mine for frequent sub-graphs amongst our tweet graphs, and use these as features in our classifier. Our results produce F1 scores of between 0.65 and 0.77, providing an improvement between 0.01 and 0.04 against the current state of the art.

In our final empirical chapter, we look to answer our third research question; *How can we detect important life events from other social media sites, such as Instagram?.* We ask

this question, as we believe Instagram to be a preferred environment to share personal life events. In this chapter, we extract a new dataset, targeting the following events: Getting Married, Having Children, Starting School, Graduation, and Buying a House. Our results find that our methodology provides F1 scores between 0.78, and 0.82, an improvement in F1 score between 0.01 and 0.04 against the current state of the art.

Contents

Listing of figures

For somehow putting up with me throwing away a lucrative career in London to pursue a PhD, this thesis is dedicated to Joanna, as well as our cats, Hazel and Willow.

Acknowledgments

Firstly I would like to thank my main supervisor, Professor Harith Alani. His advice has been invaluable throughout the course of my PhD, and has helped hone both my written, and critical thinking skills. I feel incredibly lucky to have had such a great supervisor.

Secondly, I would like to thank my other Supervisor Dr. Paul Mulholland. The numerous talks and discussions in his office have helped shape a lot of the ideas within this thesis. His help over the past five years has been warmly appreciated.

I would also like to thank Dr. Miriam Fernandez for co-authoring some of the papers supporting this thesis. I feel that I have deeply benefited from her constant optimism and feedback of my work.

Special thanks goes to the ReelLives team, including Professor Pam Briggs, Dr. Matthew Aylett, Dr. Finola Kerrigan, Dr. Lisa Thomas, Dr. Smitashree Choudhury, Andrew Hart, and Elaine Farrow. The experiences I gained from work done as part of that project has helped influence some of my thesis, as well as open my mind up to other research areas such as cognitive psychology.

In addition, I would also like to thank the Newcastle team at Lhasa. A special mention goes to Paul Sehgal, who helped support me by approving thesis writing holidays on short notice.

I would also like to thank my parents, Peter and Pauline, as well as my sister Holly, for emotionally supporting me throughout my thesis.

Finally, I would like to thank my girlfriend Joanna, who has been my main pillar of support over the course of the past five years.

1

Introduction

Since the start of the twenty first century, social media has become a ubiquitous form of communication, with companies such as Facebook[fac], Twitter[twi], and Instagram[ins], becoming every day constants in our lives. Across the decades we have been using these sites, it stands to reason that a large wealth of personal data has been recorded, and shared across different platforms that pertain to our *digital persona*[35]. However, there is a lack of tools to help sift through this wealth of content sufficiently to find those posts that mean something to us.

One immediate question that arises here is, what makes a post salient to us? We argue that one particularly useful measure are those posts which we share about our own per-

sonal life events. Thus, to that end, this thesis aims to look at the automatic identification of these posts across different forms of social media.

While social media event detection is not a new concept, the work to date on identifying personal life events is under researched. Thus, we offer the following contributions as part of this thesis:

- We look at targeting several life events that have been identified in cognitive psychology as culturally and age shared, including: Getting Married, Having Children, Death of a Parent, Starting School, Falling in Love, Graduation, and Buying a House.
- We investigate the usage of interaction, statistical, and semantic features in the detection of these types of events
- We develop a novel approach to generate semantic and syntactic graphs from social media posts, which are then mined for frequent sub-graphs to be used in supervised classifiers
- We create classifiers for two social networks: Twitter and Instagram

The rest of this chapter is outlined as follows. We first explain our motivations for detecting personal life events on social media in Section 1.1. Then in Section 1.2, we identify several challenges and gaps within the current literature. In Section 1.3, we introduce our research questions and hypothesises. Finally, we give a broad outline of the rest of this thesis in Section 1.4.

1.1 Motivation

As described in our introduction, one of our main motivations for pursuing this research is to help the identification of salient posts across our different social media platforms. Those of us with large historical social media accounts, may have over a decade worth of content,

yet we rarely explore our historical posts. Thus one of our main motivations is to help start providing tools that allow users to identify those posts that might help filter our historical social media content.

Besides identifying events as a tool for personal reflection, there are obvious benefits within marketing. A study by the Royal Mail[64] has identified that over 50% marketers see life events as a new sales opportunity to customers. This is an increase from 16% in 2014, highlighting a growing interest with the industry. Reasons given include providing an opportunity to engage with a customer, new sales opportunity, and an increased likelihood that customers might switch. There have been several studies that have shown[69,19,70] people are more likely to change buying patterns after one of these life events have occurred.

It is not hard to consider that after giving birth, an individual might be interested in buying baby products. Other events such as buying a house might require content insurance, after getting married, a couple might be interested in purchasing life insurance. This type of marketing is already employed on Facebook. For example, a professional wedding photographer has the ability to target customised adverts at recently engaged couples on Facebook[99].

Besides marketing, there are social opportunities as well. While the research in this thesis looks at identifying a specific set of events, the techniques and tools developed can be adapted to other types of life events. For instance, cyber bullying is an important growing issue[24,119,106], where young people are abused online anonymously. One of the issues with bullying is reluctance to come forward and find help, requiring intervention guides to be created to help solve this problem[23]. Adapting the research in this thesis to target a life event of being bullied online, could help enhance current intervention techniques from parents

and teaches and reduce child suffering.

Outside of life event detection, methodology developed in this thesis can be applied to other classification tasks on social media. The automatic detection of radical Twitter accounts is currently both a pressing political, and social matter. To that end, methodology produced as part of this thesis in chapters 5, and 6, have already been shown to enhance F1 scores of classifying pro/anti ISIS twitter accounts[98].

1.2 Challenges and Gaps

Until now, the vast majority of research into event detection has focussed on events such as trending topics and breaking news (e.g. Phuvipadawat & Murata[85] Sankaranarayanan et al.[101] Cataldi et al.[25] Mathioudakis & Koudas[68]), disasters (e.g. Sakaki et al.[100] Cheong & Cheong[29]), and topic retrieval(e.g. Becker et al.[16] Massoudi et al.[67] Metzler et al.[74]). In addition, nearly all relevant work has focussed on Twitter. Our suggestion for why this is owing to how easy it is to extract large datasets for processing[65], or live stream for real-time clustering tasks such as trend detection.

However, life events are not as prevalent as the aforementioned types. Most people will only experience a handful of significant events in their lives, thus the availability of such data on social media is expected to be hard to come by. For example, Di Eugenio et al[43], collected over 1 million tweets on two life events, Getting Married and Employment. After keyword and spelling filtering, Getting Married had 491 (0.049%) tweets referencing the life event, while employment had 488 (0.048%). Thus it is expected that our first major challenge will be the collection of useful datasets.

The difficulty in generating useful datasets will also likely to lead to smaller datasets.

Detection methods to date have exploited features related to the volume of availability in the dataset (e.g. bursty features[68]), or have employed clustering methodologies to group large quantities of messages together. Using these methodologies will not be applicable to smaller datasets, and will most likely need new ways of extracting information from each post. Thus, an additional challenge will be to leverage as much information as we can per post, and develop new ways that can detect events outside of traditional bag-of-word models.

Additionally, as already pointed out, the vast majority of event detection, and all life event detection (to the best of our knowledge), has been focussed on Twitter. A study by the American Press institute[94] surveyed 1000 twitter users for their use cases. Only three in ten users use Twitter as a platform to 'Tell others what I am doing and thinking about'. While we accept that life events will appear on Twitter, we stress the need to explore detecting life events from other additional platforms.

We would suggest Facebook as another suitable platform for locating life events on, however targeting Facebook has a high barrier for obtaining that type of data. We assume that most data on Facebook will be private, thus obtaining it would be difficult. As an alternative, we think this gap could be filled with Instagram. Sheldon and Bryant[105] have already highlighted that when it comes to documenting aspects of our lives, Instagram is preferred over Twitter. In addition, Instagram follows the same follower/followee architecture that Twitter has, with most accounts being public. This should allow for easier data collection to be used in datasets.

1.3 Research Questions & Hypothesises

Our main goal for this thesis is to look at improving the performance of identifying important life events from social media. To that end, in this section we list the following research questions:

R.Q. 1: *What features would improve the classification of important life events?*

Given that we have already identified that the current state of the art on life event detection on social media is under researched, our first question revolves around what features can we use to improve upon state of the art detection. Currently, we have seen limited usage of uni-grams [43], and topic models, with some semantics [61] to detect life events, in other words, text content based features.

Thus, our hypothesis for this research question is:

Hypothesis 1: *Interaction, Content, and Semantic features can enhance the state of the art for life event detection.*

Content and Semantic features look at exploring in more depth the effect of text content based features, while we look at interaction features as an alternative. We introduce the concept of each of these feature sets in more detail in Chapter 4.

R.Q. 2: *Can the inclusion of semantic and syntactic patterns improve performance in our life event classifiers?*

As mentioned as an earlier challenge, we can not rely on volume style features owing to the likely scenario that relevant large datasets would not be available for such life events.

Tweets themselves are already information starved. Until September 2017, a tweet could only carry 140 characters, with that number being doubled to 280. After stop word removal, and pre-processing, this leaves only a handful of useful tokens per document for text based features to use.

Thus our hypothesis for this research question then is:

Hypothesis 2: *Mined sub-graphs from expanding posts into semantic, and syntactic graphs, can enhance classification performance for detecting life events on social media.*

Our inspiration behind this hypothesis comes from areas such as cheminformatics, dealing with the classification of molecular structures[51 37 15]. In these papers, their approach looks at the extraction of frequent sub-graphs as features input into classifiers.

To use this type of methodology, we first need to look at how we can convert our posts to graphs. Our work looks at two types of graph expressions: Syntactic and Semantic.

With syntactic graphs, we look at the syntactical dependencies between tokens within a post, both spatially and using dependency parsing. Our hypothesis for these types of graphs state that graphs of the same life event will share similar syntactical patterns.

Semantic graphs take the approach of extracting semantic concepts within a post, then expanding into popular knowledge graphs such as WordNet[75] and ConceptNet[108]. Our hypothesis for semantic graphs are those posts pertaining to the same life event, will have crossover patterns after *n* expansions into these types of networks.

R.Q. 3: *Can the techniques used in R.Q.1 and R.Q.2 be used to classify life events on Instagram?*

Our final research question looks at addressing the gap in current research at only looking into detecting life events from Twitter. As we have already argued in Section 1.2, Instagram is more likely to have posts that pertain to life events. However, given Instagram is a different social media platform, we do not know how the techniques developed for Twitter will apply. For example, we intend to use an images caption as our main source of features, yet while on Twitter a post is limited to 140 characters, Instagrams is 2,200. Thus, we wish to explore generating a second dataset from Instagram, that we can use to train new classifiers using feature sets identified throughout R.Q.1 and R.Q.2.

Hypothesis 3: *The techniques developed in R.Q.1 and R.Q.2 are better at detecting life events from Instagram than our chosen baseline techniques.*

Our intention with this hypothesis is to apply the methods and approaches we developed for Twitter, to Instagram, as well as any additional methods we believe can be used to enhance our classification performance.

1.4 Outline

In this section we present an outline for the rest of the thesis, with a brief overview of each chapter.

1.4.1 Chapter 2 - Background & Related Work

In Chapter 2 we provide an overview of the current state of the art surrounding event detection, and highlight the lack of work within the area of life event detection. We describe/discuss the strengths and weaknesses of current approaches to identifying life events on social media.

1.4.2 Chapter 3 - Methodology

In Chapter 3 we present an outline of the methodology used within this thesis. We also describe in more detail some of the software produced to support Chapters 5 and 6.

1.4.3 Chapter 4 Life Event Detection on Twitter

In Chapter 4 we look at investigating R.Q.1 by expanding on the current state of the art. We start by annotating a new Twitter dataset, using CrowdFlower, for the following Life Events: Getting Married, Having Children, Starting School, Death of a Parent, and Falling in Love. We look at extending work done by Eugenio et al [43] by including bi-grams, and tri-grams within our features, as well as the inclusion of content metrics, and semantic features. In addition, we investigate the use of interaction features for life event posts, to see if those about life events attract a higher number of interactions. We demonstrate a significant, but minor improvement over our baseline.

1.4.4 Chapter 5 - Frequent Sub-Graph Mining to Detect Personal Life Events on Twitter

In Chapter 5 we address R.Q.2 by developing several graph models: Token Graphs, Dependency Graphs, WordNet Graphs, and ConceptNet Graphs. Given these models, we then convert our dataset generated in Chapter 4 into each type of graph, and use frequent sub-graph mining to extract frequent patterns to be used in our classifier. We demonstrate a significant improvement over our chosen baselines, with an average F1 of 0.73 across our events.

1.4.5 Chapter 6 - Detecting Personal Events from Instagram

In Chapter 6 we address R.Q.3 by extracting a new dataset from Instagram. We adjust our life event list to target the following events: Getting Married, Starting School, Giving Birth, Graduation, and Buying a House. Due to the number of hashtags Instagram uses, we look at how the performance of tokenising hashtags increases our overall performance, while performing several enhancements to our graph based approach from Chapter 5. When comparing against baseline methods, we see a significant improvement, and an average F1 of 0.8 across our events.

1.4.6 Chapter 7 - Discussion and Future Work

In Chapter 7 we discuss the content, work, and approach of this thesis, as well as discuss future work within this area.

1.4.7 Chapter 8 - Conclusion

In Chapter 8, we present a summary and our concluding thoughts on the work carried out within this thesis.

1.5 Publications

This section, where applicable, lists publications associated with each chapter.

1.5.1 Chapter 4

- Dickinson, T., Fernandez, M., Thomas, L. A., Mulholland, P., Briggs, P., & Alani, H. (2015a). Automatic Identification of Personal Life Events in Twitter. In *Proceed-*

ings of the ACM Web Science Conference, WebSci '15 (pp. 37:1–37:2). New York, NY, USA: ACM

- Dickinson, T., Fernandez, M., Thomas, L. A., Mulholland, P., Briggs, P., & Alani, H. (2015b). Identifying Prominent Life Events on Twitter. In *Proceedings of the 8th International Conference on Knowledge Capture*, K-CAP 2015 (pp. 4:1–4:8). New York, NY, USA: ACM

1.5.2 Chapter 5

- Dickinson, T., Fernandez, M., Thomas, L. A., Mulholland, P., Briggs, P., & Alani, H. (2016). Identifying Important Life Events from Twitter Using Semantic and Syntactic Patterns. In *WWW/Internet Conference proceedings 2016* (pp. 143–150).: IADIS Press

- Saif, H., Dickinson, T., Kastler, L., Fernandez, M., & Alani, H. (2017). A Semantic Graph-Based Approach for Radicalisation Detection on Social Media. In *ESWC 2017: The Semantic Web - Proceedings, Part I*, volume 10249 of *Lecture Notes in Computer Science* (pp. 571–587)

2

Background & Related Work

Event detection as a topic, is one that tracks back to the late nineties with its roots in topic detection and tracking[11]; a domain that looks at tracking events within traditional news media. With the advent of social networks such as Twitter though, the majority of research has instead focussed on trying to identify events online. However, to date, the vast majority of work has focussed on large scale events, such as trending topics[68,16], news stories[101,85], or natural disasters[100]. Very little has actually looked at identifying important life events from social media.

In this chapter, we will first look at some of the work in large scale event detection in Section2.1, to give an understanding as to why a lot of the approaches in the current literature

are not suitable for life event detection. In Section2.2, we shall then introduce the short body of work that is specifically around the domain of life event detection. Section2.3 will then introduce background material on frequent graph mining, which will be required later on for chapters 5 and 6.

2.1 Social Media Event Detection

In this section we discuss some of the background literature around event detection on social media. As already stated, most of this work focusses on detecting breaking news[101 85], trending topics[68 16], controversial events[87 88], natural disasters[100], or topic retrieval[16 67 74], where the methodology used is not suitable for the problem that this thesis looks at dealing with. Surveys by Madani and Boussaid[63], Atefeh and Khreich[12], and Cordeiro[31], all provide a good overview of a lot of the approaches taken. Typically, across these surveys, methodologies are categorised as Unplanned vs Planned event detection, where unplanned deal with the discovery of unknown events, and planned attempt to identify a certain type.

Most work looks at defining systems that first extract relevant posts either by clustering, or query expansion, and then attempt to summarise or extract event information from the results. Typically we see a mixture of methodologies that include unsupervised machine learning, supervised machine learning, topic models, language models, and query expansion.

Sankaranarayanan et al.[101] is such a paper that looked at identifying breaking news stories from Twitter. Their solution created a piece of software called TwitterStand, that would look at identifying breaking news stories and relating them to geographical areas. The approach identifies 'seed' tweeters who are known to publish news content, and use these as

centroids for clusters. Given their clusters, they then looked at associating tweets with various locations.

Relevant to our approach, they also used a Naive Bayes model to help separate news content from non news content using a bag of words model. In addition to this, they used two datasets to make up their corpus of tweets for the training aspect. The first was a larger static corpus of tweets that had been manually labelled as news or junk. In addition to this, they also had a smaller dynamic corpus that was taken from tweets identified in clusters. Their hypothesis was to ensure tweets were not labelled as junk that were related to current news stories, although didn't quite match the static corpus they had prepared earlier.

Phuvipadawat & Murata [85] also look at identifying breaking news. Their system is divided into a 'story finding' stage, and a 'story adjustment' stage. The work in that paper mostly focusses on story finding. For this, they use pre-defined search queries that are often included in breaking news tweets such as '#breakingnews'. These tweets are then grouped together based on the similarity of the first tweet in the group, using TF-IDF, and the top k terms. Also included is a boost enhancement that up weights scores based on the use of proper nouns. If a tweet's similarity score is greater than a specified MergeThreshold, the tweet is added to the group, otherwise a new group is formed.

Besides breaking news, general trend detection has also been a focus of event detection research on social media. Mathioudakis & Koudas [68] propose a system to detect trending topics over Twitter. Their system, TwitterMonitor, uses keyword burstiness for detecting trending terms. Given these terms, they developed an algorithm called GroupBurst that then assesses their co-occurrence within a few minutes worth of each bursty terms history of tweets. To enhance each groups description, non bursty, but frequent within each group

terms, are extracted using principal component analysis, and singular value decomposition. In addition, frequently mentioned entities per group are also extracted using GrapeVine.

Becker et al.[16] also delve into the area of trend detection. Their approach looks at first clustering events together, then given each cluster, classifying whether or not it is an event. For the clustering stage, they use a similar approach to Mathioudakis & Koudas[68], using an incremental clustering algorithm to generate clusters of posts based on similarity to existing clusters, and creating new ones if it does not fit. Once clusters have been extracted though, they then use a number of features: Temporal focussing on the volume of messages within a cluster; social focussing on number of likes/retweets/mentions; topical which assume clusters on an event will have a smaller high occurring number of keywords than non events; and Twitter centric, focussing on proportion of hashtags within the cluster. Given these features, they use a support vector machine to then classify a cluster as event or not.

Moving onto planned event detection, Popescu & Pennacchiotti[87] look at detecting controversial events from Twitter. Their approach looks at generating snapshots, where each snapshot contains a named celebrity, extracted from Wikipedia[wik], a 1-day time period, and all tweets within that time period, mentioning the entity. They use a supervised regression approach using gradient boosted trees, to identify how controversial a snapshot is. In their work they look at three models: Direct Model, two-step pipeline, and a two-step blended model. In their direct model, they look at estimating how controversial an event is in one go. With their two step pipeline they first look at using an event detection classifier to select a snapshot, then use a controversy detection regression model to estimate the controversy. In their two-step blended model, their set-up is similar to the two-step pipeline model, but also incorporates the event classifiers prediction score as a feature in their regression model.

Popescu et al. [88] then followed up on this work by looking at improving on their original method with a new classification methodology called *EventAboutness*. This methodology looks at augmenting the feature sets in their original work with additional features based on the importance of entities contained within the snapshot. Their intuition is that event snapshots will have a few important entities with additional minor entities, while non event snapshots will have multiple unimportant entities.

Sakaki et al. [100] looked at identifying earthquakes using Twitter. To identify whether a post is about an earthquake or not, their approach looks at identifying query words (e.g. earthquake, shaking). Given these query words they then have three feature categories, statistical features, keyword features, and word context features. Statistical features consider the number of words in the tweet, and the position of the query word. Keyword features consider the words within the tweet (e.g. a bag of words model), while word context features look at the words before and after the context word. They then use these features with a SVM model to train and classify tweets as being about earthquakes or not. Given these classifications, their system then looks at using these positive tweets within two models for earthquake detection. The first is a temporal model to detect spikes of activity (modelled against a Poisson distribution), to detect whether an earthquake has occurred, and a second spatial model to identify where the earthquake occurred.

Moving away from geographical indicators, Becker et al. [16] look at a query expansion system. For example, users who might be interested in events around the venue central park, might simply use 'central park' as their search query. Their work looks at mimicking such behaviour, deploying several different query strategies. Some of these include the title and venue of a location (e.g. Yoga at the Great Lawn Central Park), or the title plus location,

minus stop words (e.g. Yoga Great Lawn New York City). These strategies lead to high precision, low recall results (although the authors do not highlight numerical evidence for this). To improve recall, they then use term-frequency and co-location techniques to generate new queries from their high precision dataset.

Massoudi et al.[67] also consider query expansion, but as an extension to a language model including textual and post quality indicators. In addition, a temporal aspect is considered for post quality and query expansion, weighting more recent posts to the query higher. Textual indicators were taken from work done by Weerkamp & Rijke[120], and included looking at capitalisations, emoticons, post length, and the existence of hyperlinks. Post quality indicators included reposts, followers, and recency. Each of the quality indicator values are averaged into a global prior probability. The query expansion extension ranks the top k terms, and attempts to take into account the evolving language of a post. For example, posts closer in time to the query, may have different co-occurring keywords to the query terms, as opposed to older posts.

Finally, Metzler et al.[74] also propose a query expansion based system. Their work looks at splitting detection into two tasks: query expansion, and summarisation. In the query expansion phase, a user would produce a query, such as *earthquake*. Given this query, the system would generate N timespans (where an atomic timespan was one hour), and rank timespans based on the burstiness of the keyword within said timespan. Burstiness was calculated as the probability of the word within the timespan, against the probability of seeing the word within the whole corpus. After ranking, the K highest terms within these timespans are used as additional query words. Once the expanded query is built, the top 1000 timespans are then selected, based on using either a coverage scoring function, or burstiness

scoring function, both producing a score of the expanded query, against the timespan. In addition, contiguous time-spans are also merged. Finally, a summarisation step is applied which selects a small set of tweets from each timespan against the expanded query, using a weighted variant of the query-likelihood scoring function[86] using the burstiness of the expansion term, and a Dirichlet smoothed language modelling estimate .

As we can see from the vast majority of work in this field, a lot of the traditional approaches focus on large scale events, which can feature thousands of posts all referring to the same event. Thus the problem is many-to-one, where it looks at finding large clusters of posts associated with each event. However, personal life event detection is the opposite, where most events are expected to have a few-to-one relationship with a post, e.g., a single life event may only have a few posts referencing it. This means a lot of the approaches outlined so far are not suitable for that form of detection.

2.2 Life Event Detection on Social Media

While we highlighted some of the work done on event detection from social media in the previous section, far less has been done on the topic of life event detection. As mentioned already, there are different obstacles to the identification and detection of posts.

Thus in this section, we focus more specifically on current work in the field of life event detection. In subsection 2.2.1 we look at introducing each paper in detail, explaining the methodology used and results achieved. Then in subsection 2.2.2 we present an analysis of the work, including the approach (2.2.2.1), and results (2.2.2.2).

2.2.1 Methodologies

Eugenio et al[43] performed one of the first pieces of work in personal life event detection. Their goal was to automatically identify two types of life events: Getting Married and Employment. Their dataset is initially generated from a corpus of 1 million tweets, collected over several weeks in three hour intervals. Their tweets are limited to those only from the United States to increase the chance of retrieving American English tweets.

In order to filter down their corpus, they selected several key words for their two domains. For employment, they manually chose three keywords: new job, laid off, interview, job offer. For marriage, they selected keywords based on mining several domain specific sites (wedding websites), and selected the top three keywords ranked by TF-IDF[91] scores, resulting in *engaged*, *married*, and *wedding*. Once filtered, they further reduced their dataset size using two metrics: spelling/punctuation using Hunspell[77], and those with URLs. Those tweets with poor spelling, and URLs embedded, were given lower ranks and discarded.

For their training set, they divided their datasets into several different categories for experimentation. With employment, they specified four datasets: 'someone looking for employment', 'tweeter looking for/found employment', then both categories with an extra 500 negative tweets. For Getting Married, they had three sets including: 'Someone is getting married', 'Someone is getting married, plus general statements', plus a third with 500 extra negative tweets for 'Someone is Getting Married'.

Their approach looks at just using uni-grams as features against several different classification algorithms. Their top two were complement Naive Bayes, and SVMs, and thus also considered a model using a probabilistic SVM where if the probability is less than 90%, the

post is reclassified using a CNB classifier instead.

Results showed strong accuracy for their employment dataset, with scores between 0.86, and 0.902 using their combined SVM CNB approach. For getting married though, their results were slightly different with SVMs outperforming SVM + CNB across two out of the three datasets, and scores ranging between 0.731, and 0.949 (significance is not given in these results). The authors also attempted at introducing bi-grams, but to little improvement.

Li et al[61] take a slightly different approach. They look at identifying self-reporting life events. Rather than collecting tweets for specific major life events, they initially collect a large number of tweets based on congratulatory/condolence statements such as *congratulations*, *sorry for your loss*. Given these datasets, they then look at using topic models to infer life event clusters, and have a human annotator manually label those clusters that refer to a life event. Given these clusters, they then look at expanding their condolence query words to be re-used across several iterations of collection, topic modelling, and analysis. In total, they identified 42 life events.

They then look at generating a 43 class classifier (42 events and a negative class). Their feature sets include the sequence of words within the tweet, named entity tags using a twitter specific tagger by Ritter et al.[93], a dictionary of the top 40 words for each life event category in their topic model, and a context window around said dictionary word. They used a maximum entropy classifier, and reported a combination of all features as their best score with precision at 0.382, and recall at 0.487. The authors note the low scores, but attribute it to the 43 unbalanced classifier.

In addition to this, they also look at a self-reporting classifier that can identify whether

the owner of the tweet is talking about themselves. For their feature sets they include bi-grams, detecting factuality words, tense, and whether the subject of the tweet is first-person (e.g. the subject comprises of I). In addition, they include the highest probable word in that tweets event from their topic model, and include that as well as the dependency path between subject and it (if the subject is first person, and the topic word is a verb). They report an accuracy of 0.82 when all features are included, although low precision (0.51), and recall (0.48). As a final task, they also look at the extraction of content about each event from the posts, although that is outside of the scope of this thesis.

Cavalin et al[27] look at generating a multiple classifier system for the travel life event. Two datasets are constructed: one for English with 93 positive examples, and 414 negative; one for Portuguese with 138 positive, and 361 negative. The authors make no mention of how the dataset was collected except using Twitter Search API, and manually annotating the collection.

Their approach looks at building a classifier that takes as input a conversation document (post with conversation), and considers four types of feature sets: extend uni-grams, extended bi-grams, co-occurrence n-grams. Extended uni-grams, and bi-grams, looks at extracting aforementioned n-grams from each post document, including the extended conversation document. Co-occurrence features are computed by looking at the co-occurrence of term i, and j, across all documents, and generates a vector of where they appear with each other.

In addition to these features, they compare each individual feature set against a multiple classifier system (MCS). Here they classify each post against a classifier built on each feature set, and perform a majority voting of each classifiers result.

Their results show MCS outperforms all individual feature classifiers by 0.232 for their English dataset, and 0.013 for their Portuguese dataset. The authors though do not compare the performance of using a majority poll system against simply combining their features into a single classifier.

Given the imbalance of the dataset in their work, Cavalin et al[26] perform further research on classifiers that can deal with highly unbalanced data, across several classes. Rather than just target travel life events, their work considers travel, wedding, birthday, birth, graduation, and death. For travel and wedding, datasets were extracted and labelled for both English and Portuguese, while the rest were only extracted for Portuguese.

Their methodology considers only bag of n-grams as a feature set as the main objective of the paper is to compare classification performance of Naive Bayes, Logistic Regression, and Nearest Neighbour, using two over-sampling techniques: random over-sampling (ROS), and synthetic minority over-sampling technique (SMOTE)[20]. They found that Naive Bayes, and Nearest Neighbour faired much better than logistic regression on their datasets when used as is. However, after over-sampling was applied, Logistic Regressions performance improved against all cases. However, F1 score across these events were still relatively low at below 0.5 in most cases.

2.2.2 Analysis

Given we have now presented the state of the field, we can now compare and analyse the approaches that have been taken by others.

Table 2.1: Life Event Detection Approach Overview

Paper	Life Events	Objective	Methodology
Eugenio et al. [43]	Getting Married, Employment	Supervised ILEs	ML - Binary Classification
Li et al. [61]	41 Discovered Life Events	Discovery & MLEs	ML - Topic Models, Classification, Entity Extraction
Cavalin et al. [27]	Travel	Supervised ILEs	ML - Binary Classification
Cavalin & Cavalin [26]	Travel, Wedding, Birthday, Birth, Graduation, Death	Supervised ILEs	ML - Binary Classification

2.2.2.1 Objective & Methodology

First, let us consider the objective and methodology used for each of the four papers presented on life event detection. Besides looking at classifying life events, each author has a slightly different objective in mind. Table 2.1 shows each paper, the type of life events used in the work, and the approach taken where ILE is individual life event, MLE is multiple life events, and ML is machine learning.

First, we can see that all papers to date have looked at extracting life events from social media using a machine learning approach involving at least some form of classification model. The most similar in objectives are Eugenio et al. [43], Cavalin et al. [27], and Cavalin & Cavalin [26], where each define a pre-set list of life events they wish to classify, and use a supervised classification approach to do so.

Li et al. [61] takes a slightly different approach where rather than look to classify a pre-set list of life events, they look at first discovering via topic models, life events that occur on Twitter, then use this as the basis of a pipeline to extract properties for self reporting life events. Additionally, they are the only paper that uses any form of unsupervised machine

learning (topic models) in their approach.

These two approaches could be categorised in a similar way in the previous Section(2.1) where we showed planned (supervised ILE) versus unplanned (discovery MLE) event detection.

2.2.2.2 Results

In this section we provide an analysis on the four papers we have highlighted in the section, and start by suggesting that a direct comparison of the results is not possible.

While we categorised our papers into Supervised ILE and Discovery MLE in the previous section, no two papers perform the same task.

Eugenio et al. [43] consider a number of different permutations of their collected dataset, and focus on reporting the best type of classifier for each dataset. They report high accuracy, but as this is the only metric they report on an imbalanced dataset, it makes it difficult to compare against other approaches without re-running the experiment. For example, a 90/10 dataset split can still retain 90% accuracy by simply classifying everything as the major class).

While Cavalin et al. [27] report precision, recall, and F1, their main objective is to compare the performance of a multiple classifier system against the Travel event, which will have a completely different dataset to that of Eugenio et al. [43]. In their follow up work Cavalin & Cavalin [26] consider a different objective where they compare the sampling approach to help build better classifiers for imbalanced datasets. While both papers look at a Travel event classifier, the F1 score decreases from 74.8 in their first paper, to 43 in their second. However, this is because the goal of the second paper is to consider a highly imbalanced dataset

of 3.9%. In comparison, their first paper had a class imbalance of 22.4%.

When we consider the Life Event Classifier in Li et al. [61] pipeline approach, we see the worst reported results with an F1 of 42.8. However, even though they employ a similar classification approach to the other three papers, theirs is a 43-way labelled classifier which is a far more difficult challenge than a simple event specific binary classifier.

2.3 Frequent Graph Mining

As background material for chapters 5 and 6, in this section we will introduce the concept of frequent sub-graph mining. The goal of this technique is to detect and discover all sub-graphs within an item set that occur above a specified count. Before explaining this in more detail, we present several items of vocabulary that are commonly used in the literature.

Itemset This refers to the list of graphs that we mine frequent sub-graphs from. For our use case, this is the list of graphs generated from our training set for our classifier, with a one-to-one relationship between graph and post.

Sub-graph A sub-graph in this sense is a connected graph of nodes and edges, found within a super-graph.

Support We refer to support as the the number of times a sub-graph occurs across an item set.

Minimum Support The minimum support is specified at the time of mining, and is the minimum number of times a sub-graph must appear within an item set before it can be

considered *frequent*. This can be expressed as either an absolute value (e.g., 10), or a relative value (e.g., 10% of graphs).

For the purposes of this thesis, we shall be using a framework called ParSeMiS developed by Philippsen et al[84]. ParSeMiS is a parallel and sequential mining suite, with several popular algorithms: gSpan[126], CloseGraph[127], Gaston[79]. In addition to this, the library supports parallel processing, allowing us to use multiple threads to speed up the mining process.

2.3.1 Graph Mining Algorithms

One of the most popular sub-graph mining algorithm is gSpan developed by Yan and Han[126]. Traditional approaches to sub-graph mining, such as AGM[ino], and FSG[60], involved using apriori[10] based candidate generation. This poses a major challenge to sub-graph mining, due to sub-graph isomorphism being an NP-Complete problem[30]. Apriori approaches will discover all frequent sub-graphs at node count k, then generate all possible candidates for $k + 1$, by combining all patterns at k, and known frequent sub-patterns at 1. However, this can generate a large set of candidates, and due to the challenge sub-graph isomorphism, pruning false positives is costly. gSpan however does not use the apriori principal in mining, and instead builds a lexicographic order amongst graphs, and maps each graph to a minimum DFS code. Depth first search[110] is then applied to mine frequently connected sub-graphs efficiently. An example of performance increase is shown where FSG took 10 minutes to mine a chemical compound dataset with 6.5% minimum support, where gSpan took 10 seconds.

Yan and Han followed up their work on gSpan with CloseGraph[127]. While implemented in a similar way to gSpan, an important pruning action is taken when identifying frequent

sub-graphs. First we must understand the concept of a closed frequent itemset. An itemset can be considered closed, if there exists no parent with the same support. For example, let us say we mined three frequent sub-graphs, A, B, and $(A) - (B)$. If all their supports were the same, we would say that A, and B are closed sub-graphs, and ignore $(A) - (B)$, as that particular sub-graph only ever occurs with A, and B. This pruning mechanism is what is used with CloseGraph.

Gaston was developed by Nijssen and Kok[79], and searches for frequent graphs via three steps. First, it searches for frequent paths within the dataset, then frequent free tees (paths without cycles), and finally cyclic graphs. Their results reported faster run times than gSpan, although comparisons were not made against CloseGraph.

2.4 Classifier Algorithms

In this section we present a brief overview of the classification algorithms used throughout this thesis.

2.4.1 NaiveBayes

NaiveBayes[76] is a probabilistic classifier that uses Bayes theorem[109] (shown below) to calculate the probability of a class, based on the probability of a feature existing within the given class.

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

The naive aspect of the classifier is the assumption of conditional independence between each feature given the class value. Even with its simplicity, compared to other more complex

classifiers, NaiveBayes has been shown to perform relatively well, specifically with document classification[72] and spam filtering[73]. It can also perform well with small amounts of training data[128].

2.4.2 J48 Decision Tree

J48[124] is a Java version the of C4.5[90] implementation of ID3[89]. Decision trees work by utilising the concepts of Information Gain and Entropy.

Entropy can be viewed as the level of impurity within a group of attributes, i.e., a measure of how biased an attribute is to a particular label in our dataset.

Mathematically, this can be defined as

$$E(X) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

where X is an attribute/class, c is number of classes, and p_i is the frequentist probability of X given c.

The possible value of E is between 0 and 1, where a low value of E represents a high level of purity (i.e., 0 where all attributes bias towards one class), whereas a high value represents impurity (i.e., 1 where all attributes are split evenly between all classes).

Information gain is a metric in the reduction of uncertainty given an attribute. Mathematically this is defined as

$$IG(Y, X) = E(Y) - E(Y|X)$$

With entropy and information gain defined, an overview of a decision tree can be given.

Given a list of attributes and a number of classes, a decision tree will create a tree of the attributes and their values, with leaf nodes representing a predicted class. Information Gain and Entropy are used to recursively build the tree and select which next decision would return the highest amount of information gained.

When splitting the tree the weighted averaged entropy is taken for a class given the attributes value and used to calculate the information gain for the class given the attribute.

To help prevent over-fitting, decision trees can also be pruned in order to remove sections of the tree that provide little information.

2.4.3 Random Forest

A Random Forest[21] is an ensemble learning method that uses multiple decision trees to predict a class.

Given our description of decision trees in the previous Section(2.4.2), a Random Forest will create a specified number of decision trees on various sub-samples in the dataset. The class prediction is then the mode of the class predicted by the ensemble of trees.

Typically, Random Forests will outperform a single decision tree due to the diversity of the information, and are harder to overfit.

2.4.4 LibLINEAR

LibLINEAR[44] is an implementation of a linear Support Vector Machine[32] (SVM) that is optimised only for linear classification problems. Linear SVMs will create a linear hyperplane between classes with a separation boundary often referred to as the *cost*.

A high value of *cost* will attempt to minimise the separation of the hyperplane between

itself and the features in an attempt to reduce the misclassification error, while a low value will create a larger boundary leading to a higher number of misclassifications.

Typically, the cost value can be optimised to make a trade off and help reduce overfitting within the classifier.

2.5 Conclusion

In this background chapter, we have given an overview of the state of event detection on social media, and highlighted how the majority of work in the area has mostly dealt with large scale events, with most approaches not being suitable for the identification of personal life events. Those works within the domain of personal life event classification look at the construction of a classifier. In addition, we have given a brief background overview of frequent sub-graph mining, which is applied in chapters5 and6.

3

Methodology

In this chapter, we present the approach taken throughout this thesis in order to address our research questions. First, we are treating this as a supervised classification problem[96]. Supervised classification is a branch of machine learning that looks at the use of a labelled training set to make further labelled predictions.

Figure 3.1 shows an overview of our approach in this task.

As we can see, initially, our approach looks at extracting datasets from both Twitter, and Instagram. Given these datasets, we then look at annotating these using CrowdFlower[cro] (details in Section 4.3.2 and 6.2.3). Chapters 4, 5, and 6, then look at different ways we can extract and use features to provide the best possible classifier.

Figure 3.1: Methodology Outline

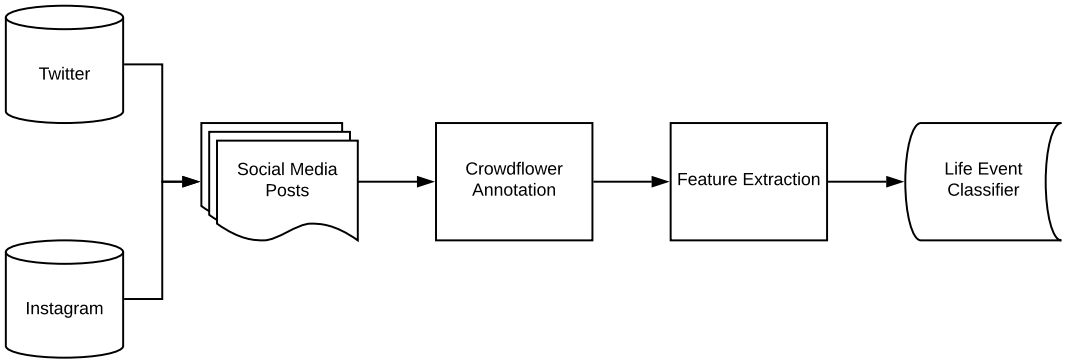


Table 3.1: Target Life Events

Chapter	Life Events
Chapter 4, Chapter 5	Getting Married Having Children Starting School Falling in Love Death of a Parent
Chapter 6	Getting Married Having Children Starting School Buying a House Graduation

3.1 Life Events

As mentioned earlier in Section 1.2, we intend to look at using cognitive psychology as a source for our life events. Work done by Steve and Janssen[54] provides us with a culturally accepted list of important life events that most people experience. More on this work is discussed in Section 4.2. Table 3.1 shows the list of life events we attempt to identify across each chapter.

3.2 Machine Learning Framework

For our experiments, we intend to use WEKA[123] as our main machine learning tool kit. There are a number of alternatives, but WEKA's experimenter is useful when we need to compare a number of different feature sets and classifiers against each other, owing to its ability to report significance (mentioned in Section 3.4).

3.3 Evaluation

Typically in machine learning, there are two types of testing methods: N cross-fold validation, or train/test split. There exist others, but typically they are subsets of these two.

N cross-fold validation takes a training set, and splits it into N equal sections. N - 1 sections are then used for training, while 1 section is used for testing. This is repeated N times until all sections have acted as a test set. The average statistical results for this are then reported to the user.

Training/test split methodology looks at providing a split of training and test data, using the training section to train our classifier, while the test section is used exclusively for testing. This is useful for large datasets, however, as we have already pointed out our datasets are unlikely to be large.

Because of this, most of our evaluation will be done using cross-fold validation.

In the majority of cases, we will also be looking at reporting three metrics: Precision, Recall, and F1[123].

We define precision as:

$$P = \frac{T_p}{T_p + F_p}$$

Where T_p are the number of true positives, and F_p are the number of false positives. Precision is an indicator of the proportion of correct classifications out of those made. A value of 1 would indicate all values in a class were correct, where as 0 would indicate non were correct.

Recall is defined as:

$$R = \frac{T_p}{T_p + F_n}$$

Where F_n are the number of false negatives. Recall measures the overall number of correct classifications made relative to the dataset. A value of 1 would indicate all possible correct classifications were made, where 0 indicates none were made.

Finally, F1 is defined as:

$$F1 = 2\frac{P * R}{P + R}$$

F1 is the harmonic mean of our precision and recall values. A value towards 1 suggests high classification performance, where as closer to 0 suggests very poor classification performance.

3.4 Hypothesis Testing & Baselines

Considering the size of our datasets, our approach is to report *P*-values using paired student t-tests.

We suggest a compatibility (or incompatibility) with the corresponding hypothesis, and include *P*-values in the results where possible to allow readers to make their own decision.

In all chapters, we provide at least one baseline approach to compare against. Due to when each chapter's empirical work was performed, some have fewer baselines than others.

3.4.1 Feature Evaluation

In addition to baselines, we employ two forms of baseline evaluations for the assessment of new feature sets and classifiers.

ZeroR simply assigns the class with the highest majority, and uses that as its prediction in any future classification task. It does not use the feature set in any way, thus makes a useful classifier to evaluate if a particular feature set has predictive power.

OneR on the other hand selects the best rule from a set of features, and then uses this as its prediction function. Typically this is a useful indicator as to judge whether a particular classifier algorithm is performing well, or whether it is relying on a single feature.

3.5 Graph API

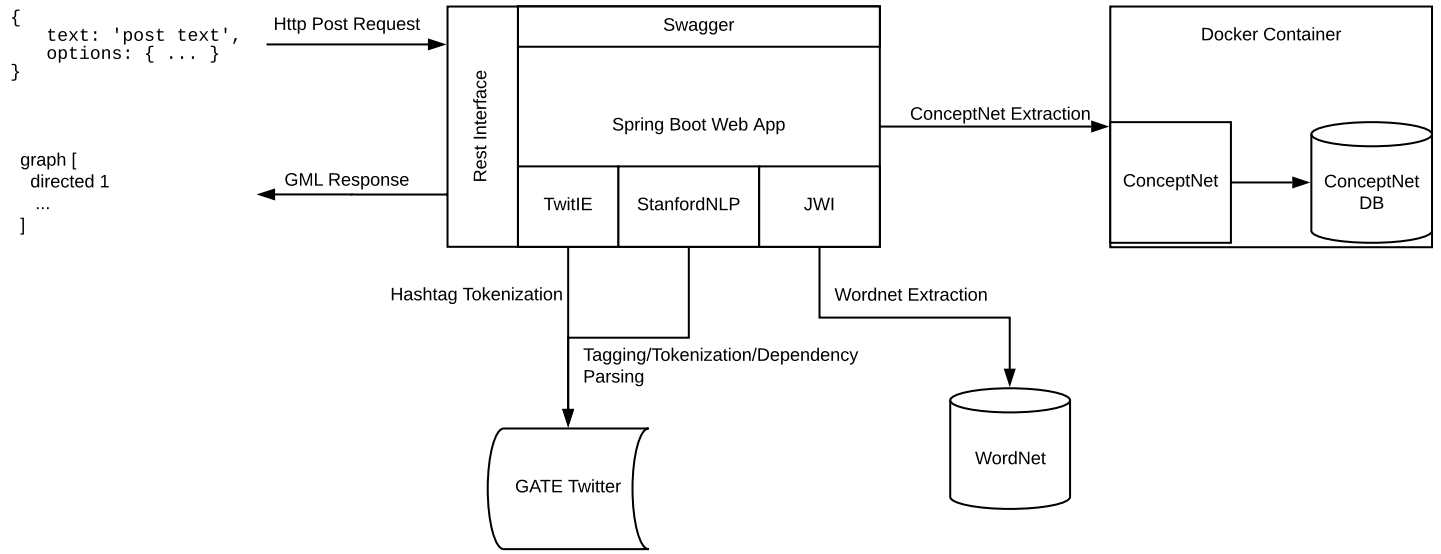
In this section we present a more detailed overview of the graph expansion system that we develop as part of chapters 5 and 6.

3.5.1 Architecture

Figure 3.2 gives a high level over view of the software built.

The core of our system is built as a spring boot application with RESTful interfaces, so any software with HTTP capability can generate graphs using it.

Figure 3.2: Graph Generation Architecture



We expose a single endpoint, *http://server-address/api/graph/create*, allowing users to POST Http requests with a JSON content body. After processing, the endpoint returns a graph in GML[50], that can be processed by most graph libraries (we use NetworkX[38] for our experiments).

3.5.2 Syntactic Graph Extraction

For syntactic graph extraction, we utilise a mixture of StanfordNLP, and GATE (which itself uses Stanford NLP). We utilise GATE's Twitter POS model[36], which is loaded into a Stanford NLP MaxentTagger. For dependency relationships, we use DependencyParser class from Stanford NLPs suite of tools, using just its default model. Ideally we would have preferred to have used something like TweeboParser[56], however the implementation of that into our chosen architecture proved to be problematic, thus we opted for an out of the box dependency parser from StanfordNLP.

3.5.3 WordNet Graph Extraction

For WordNet extraction, we used JWI[45] for WordNet extraction. Initially, we tokenise our text using the strategy in section 3.5.2 and iterate through each to see if it exists in any synsets in WordNet. To optimise matching, we also check each token with its associated POS. For every token, we return a list of synsets, which are then recursively expanded to each specified wordnet relation depth (hypernym, hyponym, meronym, similarTo).

3.5.4 ConceptNet Graph Extraction

For ConceptNet, we use a local docker image of ConceptNet 5.5, for our ConceptNet server. Similar to WordNet, we iterate over our tokenised text data, looking for possible concept candidates. ConceptNets defines its concepts using the following schema:

/c/en/concept

where we specify /en/ to signify an english concept. For example, if we had the token *wedding*, our request to ConceptNet would be *http://conceptnet/c/en/wedding*. If a token exists within ConceptNet, we then generate a graph to *n* depth for each.

3.5.5 Hashtag Tokenisation

For Hashtag Tokenisation, we use TwitIE, a GATE plugin for tweet content. This is loaded as part of the Spring boot application on startup. We first detect all hashtags within a post by tokenising our text, and checking for all tokens that start with #, being of length greater than 1. Given a hashtag, we then pass it to TwitIE, which then tokenises it using a neural network.

3.5.6 Stemming

For stemming, we opted to use PorterStemmer[116].

3.5.7 Web Client

In addition to using the REST API to generate graphs, we built a small web interface that can directly interface with our application to check the composition of the graphs. The

client is built using Angular.io[42], and uses D3.js[129] to display the graphs. Our uses case for this was to allow us to visualise the graph generation to discover any bugs within our code. Figure 3.3 shows an example screenshot of the application.

3.5.8 Stopwords

Table 3.2 shows our list of stopwords used in chapters 4, 5, and 6

Figure 3.3: Web Client

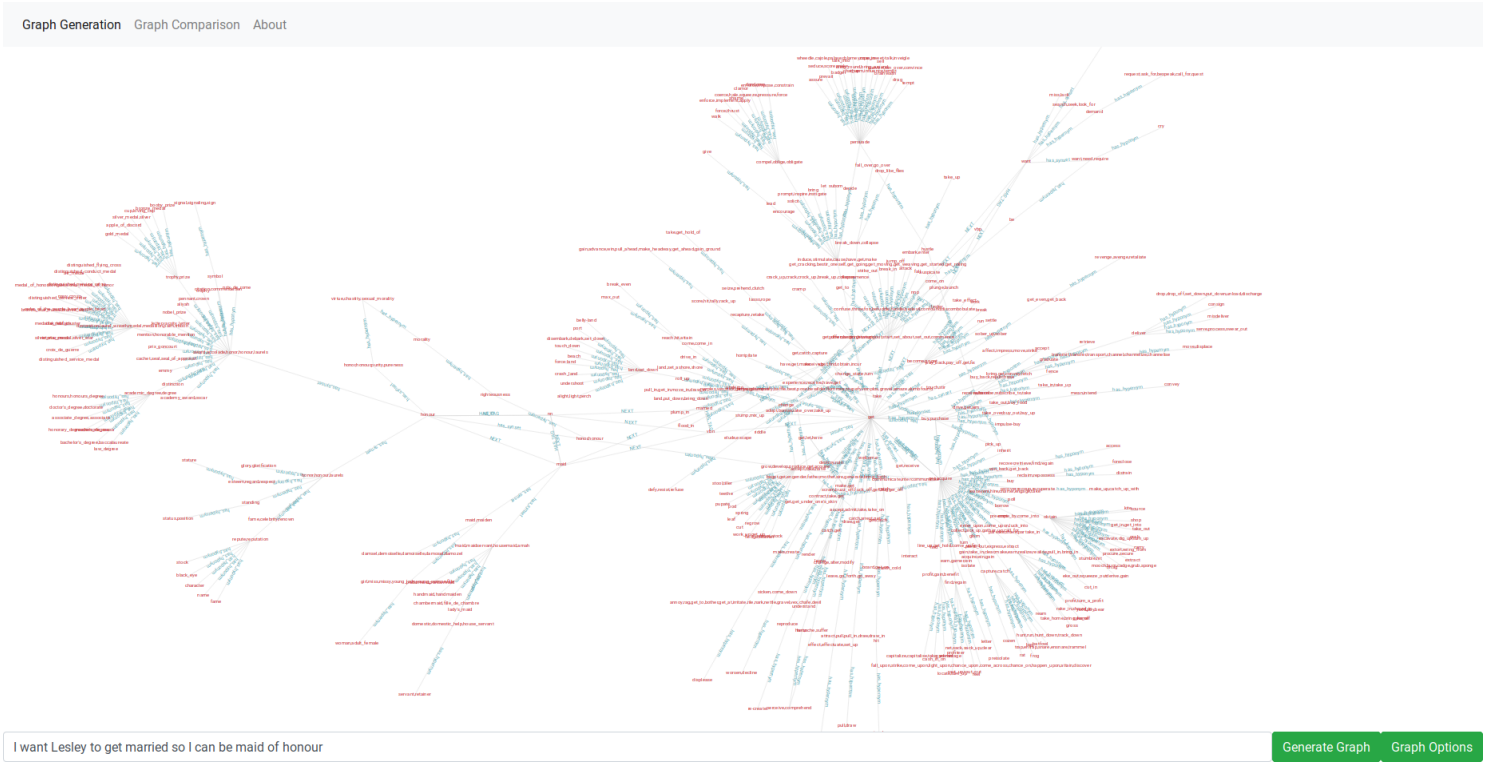


Table 3.2: Stopwords

about	d	her	mustn't	she's	ve
above	did	here	my	should	very
after	didn	hers	myself	should've	was
again	didn't	herself	needn	shouldn	wasn
against	do	him	needn't	shouldn't	wasn't
ain	does	himself	no	so	we
all	doesn	his	nor	some	were
am	doesn't	how	not	such	weren
an	doing	i	now	t	weren't
and	don	if	o	than	what
any	don't	in	of	that	when
are	down	into	off	that'll	where
aren	during	is	on	the	which
aren't	each	isn	once	their	while
as	few	isn't	only	theirs	who
at	for	it	or	them	whom
be	from	it's	other	themselves	why
because	further	its	our	then	will
been	had	itself	ours	there	with
before	hadn	just	ourselves	these	won
being	hadn't	ll	out	they	won't
below	has	m	over	this	wouldn
between	hasn	ma	own	those	wouldn't
both	hasn't	me	re	through	y
but	have	mightn	s	to	you
by	haven	mightn't	same	too	you'd
can	haven't	more	shan	under	you'll
couldn	having	most	shan't	until	you're
couldn't	he	mustn	she	up	you've

4

Detecting Personal Life Events on Twitter

In this chapter we extract a new gold-standard Twitter dataset, and construct a classifier that labels whether or not a specific type of life event has occurred in the tweet. We look at comparing a current state of the art baseline (Uni-grams Eugenio et al. [43]), against three new feature sets: Content Features, Semantic Features, and Interaction Features. Our conclusion demonstrates that a combination of these feature sets can outperform current state of the art techniques on our collected dataset.

4.1 Introduction

As discussed in section 1.2, the vast majority of work done in social media event detection has focused on large scale events. Only a small amount has been attempted in recent years to look at identifying smaller infrequent events.

One of the hurdles in these types of challenges is dealing with smaller datasets. Due to how infrequently these types of events occur on people's timelines, collecting enough to perform accurate event detection using techniques such as Sakaki et al. [100], Culotta [33], or Phuvipadawat & Murata [85] are unlikely to work well. Instead, we have to focus on feature sets that work well on smaller datasets.

Current state of the art has so far focussed on mostly features associated with the text of the tweets. In this chapter, we explore extending some of this work, as well as looking at a non content based feature set: Interaction features.

Thus this chapter focusses on our first research question detailed in section 1.3:

R.Q. 1: *What features would improve the classification of important life events?*

To this end, we look at proposing several new feature sets to identify personal life events on Twitter: content, semantic, and interaction. For content features we look to extend the work done by Eugenio et al. [43] considering uni-grams, bi-grams and tri-grams, as well as additional content metrics used in Rowe & Alani [95]. With semantic features, we hypothesise that while text might differ between posts, a life event will share the same similar semantic space. Finally, with interactions we intend to see whether certain life events have a similar set of interactions. For example, a wedding may have an usually high number of likes and comments, while people are unlikely to like a post about the death of a parent.

In order to test our feature sets, we also look at extracting and annotating a new dataset from Twitter for five different life events identified and inspired by work done in cognitive psychology: Having Children, Starting School, Getting Married, Falling in Love, and Death of a Parent. We collect a dataset for each event from Twitter to use to extract features as input to our classification models.

Our results show that a combination of content, and semantic features out-performs a uni-gram model by a 0.003-0.01 increase in f-measure. While we see a compatibility with our hypothesis, this improvement is negligible. For interaction features, we discover several features that correlate to events and help boost other feature sets, specifically in precision.

The rest of this chapter is outlined as followed. In section 4.2 we discuss our definition of a personal live event, and how we select them. Section 4.3 shows how we collect and annotate a Twitter dataset for each of the personal live events we select in section 4.2. We then introduce our proposed feature sets in section 4.4, followed by explaining how we generate each of our feature sets in section 4.5. Section 4.6 covers our evaluation set up, with section 4.7 reporting the results of our evaluation. Finally in section 4.8 we discuss the outcome of our research, and present a summary of the work done in section 4.10.

4.2 Personal Life Events

Before we can start extracting datasets, we first need to consider what a personal life event is. For our work, we turn to cognitive psychology, which has a history of manually identifying personal life events, typically within the realm of autobiographical memory[122]. Autobiographical memory is a memory system that can be considered a special type of episodic memory[114]. Episodic memory deals with remembering personal experiences, as opposed to

semantic memory which deals with general knowledge. Most episodes are assigned to short term memory and forgotten (e.g. what I had for lunch yesterday). Autobiographical memory though is more long term, and deals with important markers in our lives, e.g. when we get married, jobs, or deaths.

Related to autobiographical memory, are cultural life scripts[17]. Scripts within cognitive psychology deal with our understanding of how events play out. Cultural scripts refer to our shared representation of timing of major transitional life events.

Janssen & Rubin[54] asked participants in a study of age ranges between 16 and 75, to identify life events that would occur to a fictional baby over the course of its life. One aim of the study was to consider whether there was shared consensus between those age groups who had experienced these events, versus those who had not. The results showed this was in fact true, and produced as part of it a list of 48 types of life events.

From this list, we have decided to target the top five events identified: *Having Children, Starting School, Getting Married, Falling in Love, and Death of a Parent*.

4.3 Dataset

In this section we present how we extracted and annotated our gold standard dataset to perform our evaluations against.

4.3.1 Collection

To extract our dataset, we use keywords, similar to Eugenio et al. [43] in their work.

Our approach looked at splitting each event into a minimum number of concepts, and then use WordNet to find related terms. Table 4.1 shows our root concepts for each theme,

Table 4.1: Root Concepts used for Keywords

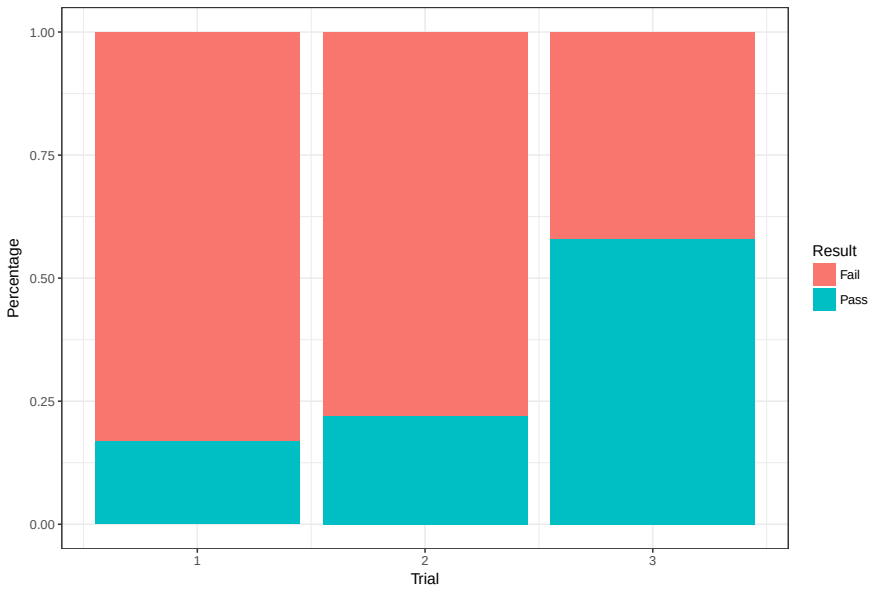
Life Event	Concepts	Keywords
Having Children	child	baby, infant, child, kid, minor, nipper, small fry, toddler, tike, tyke, nestling, fry, family
	birth	birth, born, bear, deliver
Getting Married	marriage	marriage, marrying, wedding, marriage ceremony, get married, hook up with, espouce, tie the knot, ring bearer
Death of a Parent	death	die, decease, perish, pass away, expired, kick the bucket, conk, give-up the ghost, drop dead, pop off, choke, croaked, snuffed it
	parent	parent, papa, father, dad, dada, daddy, pa, papa, pappa, pop, mother, ma, mamma, mom, momma, mommy, mammy, mum, mummy,
Starting School	school	school, crammer, conservatory, alma mater, reception, kindergarten, primary school, secondary school
	start	begin, start, first
Falling in Love	love	love, loving
	partner	him, her, you, girlfriend, girl, lady friend, baby, bay cakes, ball and chain, bby, bbygrl, better half, bf, bitch, boo, boyfriend, flame, get together, main squeeze, old lady, old man, pookie, sweetie, trophy wife

which we then extracted synsets for, and used those as our keywords. We also included the different tenses for each word using the website Verbix[ver], including those in our query as well.

In order to collect more tweets outside of the 7 day search window for Twitter, we then developed an extraction tool in python* to leverage Twitters new tweet indexing policy[115]. total of 3608500 tweets were collected.

*https://github.com/tomkdickinson/Twitter-Search-API-Python

Figure 4.1: Quiz Funnel Pass Rate



4.3.2 CrowdFlower Analysis

To annotate our dataset, we used CrowdFlower[cro]. CrowdFlower is an online Crowd sourcing website for annotations, where annotators sign up for tasks, and are paid a sum per batch of annotations they do.

Rather than have *n* annotators annotate the entire dataset, CrowdFlower batches up the job and ensure each unit is annotated by three different people.

For each experiment, a user will have to go through a short test to ensure they obtain a high enough score in order to proceed.

In order to obtain decent annotations, we experimented with several small trials using different types of questions to see which worked. useful feature of CrowdFlower is a trial quiz that annotators must take before they can proceed with further annotation jobs. Figure 4.1 shows the results for each trial with its pass rate.

Table 4.2: CrowdFlower Questions & Agreement Trial 1

Question Name	Agreement
Is this tweet about an event?	81.31%
Was the tweet tweeted before, during, or after the event?	48.38%
Is the author of the tweet experiencing the event?	50.1%
Is anyone else named in the tweet experiencing the event with the author?	35.52%
Is anyone else named in the tweet experiencing the event?	23.73%
Did the event happen where the tweet was tweeted?	44.74%
Is this tweet about [event_type]?	71.68%

Table 4.3: CrowdFlower Questions & Agreement Trial 2

Question Name	Agreement
Is this tweets theme about [Event Name]	95.98
Is this tweet about an event?	74.96

Trial 1 looked at asking a number of questions about each tweet. Table 4.2 shows each question and its agreement rating.

As can be seen, we saw low agreement ratings on a number of our questions, and had to stop after only 254 judgements as only 17% of people were making it through our quiz funnel.

For our trial 2 we reduced the number of questions to just two. Table 4.3 shows the questions and associated agreement rating for each. After 316 judgements however, we still saw far too many people leaving the quiz. Much of our feedback was related to the subjectivity of whether a tweet was about an *event* or not. Because of this, only 22% of people made it through our trial quiz, and most dropped off after about one round of annotations.

Finally, for trial 3, we adjusted our second question to only annotate whether a tweet was about an *important life event*. Table 4.4 shows each question and its associated agreement rating, showing a high agreement rate for both questions (89% and 87%, respectively). For

Table 4.4: CrowdFlower Questions & Agreement Trial 3

Question Name	Agreement
Is this tweet related to [Event Name]	89.5
Is th is tweet about an Important Life Event?	87.17

this trial we obtained 47974 judgements and saw a 57% pass rate for our trial quiz.

Figure 4.2 shows our final distribution for each event. We categorise a post as being *About Event* when both questions for trial 3 were positive, and *About Theme* when only *Is this tweet related to [Event Name]* was answered positively.

4.3.3 Interaction Data Collection

In addition to collecting our tweets, we also collected interactions for each user within our annotated dataset. These were collected across all of their timelines by running the following query:

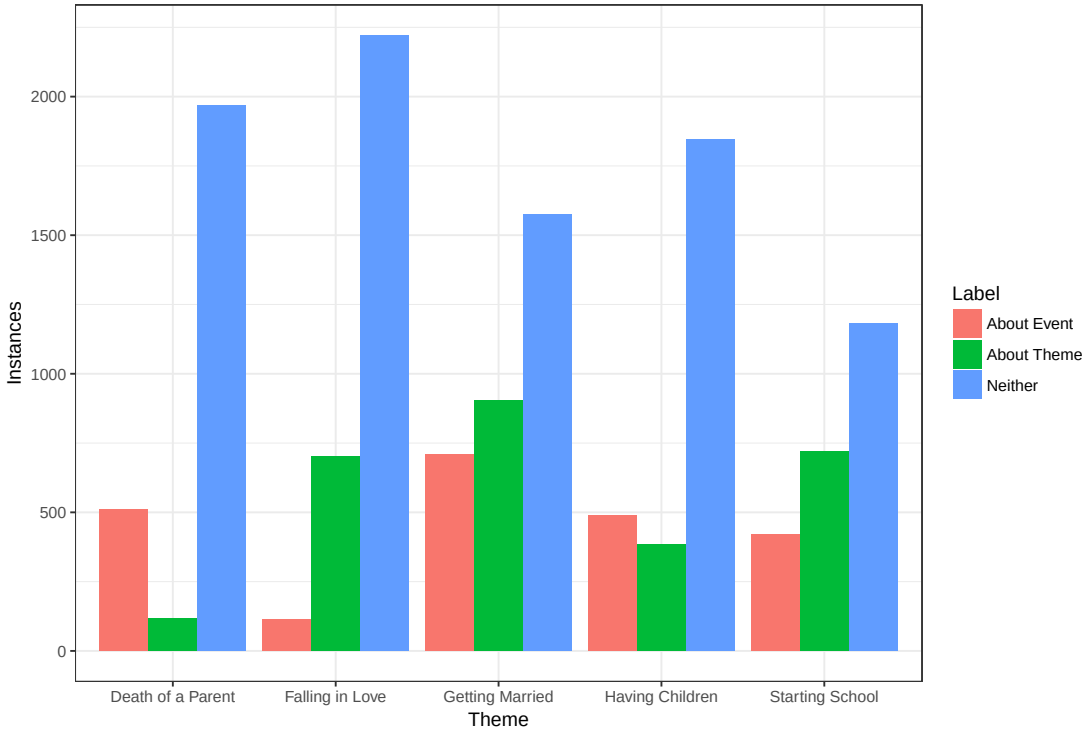
from:@screenname

We extracted timeline interaction data for 12,790 unique users, including who commented on each tweet, total likes, and total comments.

4.4 Feature Sets

In this section we introduce the feature sets we want to compare against our chosen baselines. This includes what each feature set is designed to do, and how they are extracted.

Figure 4.2: Dataset Distribution



4.4.1 Content Features

The first new feature set we are going to consider is an extension of uni-grams used by Eugenio et al [43], where we shall consider a combination of uni-grams, bi-grams, and tri-grams.

In their paper they reported an attempt to use bi-grams with no real added success, however they did not make it clear whether they tried bi-grams by themselves, or included them with their uni-gram models. Also, their work only considered the events *Weddings*, and *Employment*. We may find other themes benefit from the inclusion of bi-grams and tri-grams.

We also look at enhancing these features with additional statistical content metrics. These features are taken from work done by Rowe & Alani [95].

Length: Certain life events may have similar patterns of expression, and thus similar lengths in tweet size. Thus for this feature, we intend to include the number of words used to express a tweet.

Complexity: Complexity is a measure of token entropy within a tweet, representing how many unique terms are used to express something. A complexity of 1 would indicate that the same word is used within a tweet, while a complexity of 0 would suggest every word is different.

Let n be the number of unique terms within the post p, and f is the frequency of term t within p. The complexity can then be calculated as follows:

$$\frac{1}{n}\sum_{i=1}^{n} f_i(log(n) - log(f_i))$$

Readability: Gunning fog index[48] using average sentence length (ASL) and the percentage of complex words (PCW), i.e., those with 3 or more syllables: $0.4(ASL + PCW)$ This feature gauges how hard the post is to parse by humans, with the assumption that words with more syllables are harder to read.

Referral Count: The count of the number of hyperlinks contained within the post. Users may provide links to additional references, or non personal event posts may be linking to news stories.

Informativeness: The novelty of the posts terms with respect to other posts. We derive this measure using the Term Frequency-Inverse Document Frequent (TF-IDF)[91] measure. We hypothesise that posts about certain life events will contain terminology that is outside of the *platform vocabulary*.

$$\sum_{t \quad p} tf_{t,p} \times idf_t$$

Our sample of *platform vocabulary* is taken from Twitters sample endpoint, where we sampled 50k random tweets.

Sentiment: Assesses the average sentiment of the post (positive, neutral or negative) using SentiStrength[111]. Posts about important life events may be associated with stronger (positive or negative) sentiment.

Table 4.5: Number of entities and concepts found per theme

Theme	# Entities	# Concepts
Starting School	39	60
Having Children	124	90
Falling in Love	80	86
Death of a Parent	87	82
Getting Married	154	93

4.4.2 Semantic Features

For our semantic features, we extract two types of features: Entities, and Concepts. Entities refer to named entities[104] found within a tweet which we represent with its Wikipedia[wik] URL. Examples collected from our corpus include: http://en.wikipedia.org/wiki/MasterCard, http://en.wikipedia.org/wiki/Breastfeeding , http:// en.wikipedia.org/wiki/High_school.

Concepts are the unique set of entity types, taken from its rdf:type[103] tag. Examples collected from our corpus include: http://dbpedia.org/ontology/Work, http://dbpedia.org/ontology/Food, http://dbpedia.org/ontology/Hospital.

We extract our entities and concepts using the online NLP API TextRazor[tex], using their Entity endpoint.

Table 4.5 shows how many entities and concepts were extracted for each of our datasets.

We represent these features as a binary vector with 1 indicating an entity or concept is mentioned in a tweet, and 0 if not.

4.4.3 Interaction Features

So far all of our feature sets have focused on the textual content of a tweet. Interaction features instead are designed to consider the engagement of a tweet. Our hypothesis is that

those tweets that are about a personal life event, will have engagement levels outside of the users normal interactions. Here we present a list of proposed interaction features.

Conversation Time: A tweet with a longer conversation time may indicate a post is about a life event, as it triggers a longer conversation versus a standard post.

Conversation Length: Similar to the length of time for the conversation, a tweet about an event may receive a larger number of replies in a smaller time frame than non event tweets.

Number of Favourites: A tweet about a life event may garner a larger number of favourites, or likes, from their followers.

Number of Retweets: A tweet about a life event may garner a larger number of retweets from their followers.

Number of Unique Users: A tweet referencing other users might be discussing some form of important event that has happened to them, including those users.

Likelihood of a post gaining *n* favourites While the overall *n* of favourites might be indicative of a life event, that is an absolute value that has no context to a users previous number of favourites. Here we consider the probability of a user gaining *n* favourites, in contrast to all previous tweets. We calculate this using a Poisson distribution:

$$P(k\ \#\ likes) = e^{-l}\frac{l^k}{k!}$$

Where:

- k - Number of likes on a post
- l - The average number of likes on a users timeline

Likelihood of a post gaining *n* retweets Similar to our previous feature, we measure the relative value and likely hood of a users tweet gaining *n* retweets in the context of all their previous tweets.

$$P(k\ \#\ retweets) = e^{-r}\frac{r^k}{k!}$$

Where:

- k - Number of retweets on a post
- r - The average number of retweets a user gets across their timeline

4.5 Feature Creation

In this section we introduce how we generate our feature files, and the tools we use for classification.

4.5.1 Pre-Processing

Before we introduce how we generate our classification models, we first need to introduce a pre-processing strategy that is common amongst all our feature sets. For our content based

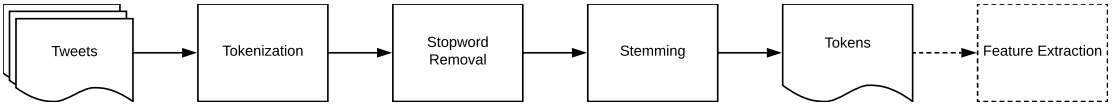


Figure 4.3: Text Processing Pipeline

features (Uni-grams, N-Grams, and Topic Models), we need to tokenize, tag, remove stop words, and stem each tweet in the same way across both our baseline and feature sets to reduce the number of independent variables that might affect the performance of each feature set.

Figure 4.3 shows the common pipeline that all tweets will pass through before it is used in feature extraction. This has been developed as a web service in Java[†] which can be interacted via REST calls to the server.

We first start by tokenising each tweet using the StanfordNLP[66] tag/tokeniser. However, rather than use their standard model for their tokeniser, we use a twitter specific[65] one developed for use in GATE[36].

Our second step is stopword removal. The stopword list we have decided to use is the one provided by NLTK[18] in python. This is included in section 3.5.8.

The final step in our pipeline is stemming. We use the standard PorterStemmer[116] as implemented as part of StanfordNLP toolkit.

4.5.2 N-Gram Extraction

To extract our n-grams, we will be utilising the text processing pipeline as illustrated in Figure 4.3 to first process and tokenize our tweet. Then, we will look at generating an n-gram vocabulary from our training sets, identifying all n-grams between one and three tokens

[†]https://github.com/tomkdickinson/social-text-graph-api.git

in length. After extraction of our n-gram vocabulary, we then generate a feature vector for each tweet, counting the number of times that n-gram appears within it.

4.5.3 ARFF Generation

In order to do this, we have written a python class that that can generate arff files for each of our desired feature sets. As part of this class, we implement two methods: *extract*, and *count*. Our *extract* method takes only our training instances, and extracts all the details for that feature set. For example, in the case of n-grams we extract our n-gram vocabulary and store that as part of our class. Our *count* method can then be ran on both our training and test instances, to generate feature vectors constructed from our previous extract method, allowing us to separate our test data from our feature discovery process.

This will only affect features such as n-grams, uni-grams, topic-models, and semantic features, as these require knowledge of the training set as a whole. Feature sets like content features, and interaction features, are both instance specific, i.e., they only need the tweet they are currently processing, and their result will not influence the rest of the dataset. This interface can be considered as being used throughout the thesis when extracting and counting any feature set.

4.6 Evaluation Setup

In this section we provide an overview of the methodology used to set-up and evaluate our experiments.

Table 4.6: Classification Parameter Tuning

Classifier	Parameter	Values
Random Forest	numIterations	1,5,10,50,100,200,300

4.6.1 State of the Art Baseline Comparison

After we have evaluated whether a particular feature set is useful or not, we then move on to comparing each against our baseline method, Eugenio et al[28]. We shall refer to this in our results as 'Uni-grams'.

4.6.2 Sampling & Hypothesis Testing

For each experiment we perform 10-Fold cross validation five times with different seeds. We then perform a paired student t-test against our baselines (as defined in section 3.4) to detect whether or not a particular combination is significantly different.

4.6.3 Classifiers

For this experiment, we have selected three different classifiers: Naive Bayes[76], J48[90], and Random Forest[21]. These are presented in more detail in sections 2.4.1, 2.4.2, and 2.4.3 respectively.

All three classifiers have been shown to work in document classification[72 14 62], and come from different categories: probabilistic, decision tree, and ensemble.

4.6.3.1 Parameter Tuning

For Random Forests, we perform hyper parameter tuning using cross fold validation. Table 4.6 shows our chosen parameters. We use the MultiSearch[92] WEKA package

Table 4.7: Dataset Composition for Thematic Classifiers

Theme	About Event	Not About Event
Having Children	489	489
Getting Married	709	709
Death of a Parent	509	509
Falling in Love	114	114
Starting School	420	420

4.6.3.2 Attribute Filters

In all cases, we normalise our attribute results between 0 and 1, using Weka's attribute filter: Normalise.

4.6.3.3 Dataset Composition

Table 4.6.3.3 shows the dataset composition. Our two classes are About Event, and Not About Event. Earlier in section 4.3.2, we presented our annotation questions. Our *About Event* class is filled with tweets that that answered positively to both of our annotation questions for their respective theme. For *Not About Event*, we use tweets that responded positively to be related to the theme, but negatively to be being about a personal life event. In order to balance our datasets with our negative class, and to avoid bias, we end up sampling independent of theme.

4.6.4 Baseline

For our baseline, we use work done by Eugenio et al. [43]. We extract uni-grams using the same process as how we extract n-grams, but limited n to 1.

While we also highlighted additional related work in sections 2.2.1, these papers were

Table 4.8: Death of a Parent Single Feature Classifier Results

Classifier	Feature Set	F1	F1 Δ	P	R	*P*-value
NaiveBayes	Eugenio	0.928	0	0.929	0.928	
	Content	0.915	-0.013	0.915	0.915	< 0.01
	Semantic	0.63	-0.298	0.65	0.638	< 0.01
	Interaction	0.423	-0.505	0.627	0.531	< 0.01
J48	Eugenio	0.916	0	0.917	0.916	
	Content	0.919	0.003	0.919	0.919	0.01
	Semantic	0.613	-0.303	0.623	0.618	< 0.01
	Interaction	0.628	-0.288	0.709	0.653	< 0.01
RandomForest	Eugenio	0.935	0	0.935	0.935	
	Content	0.925	-0.01	0.926	0.925	0.116
	Semantic	0.621	-0.314	0.631	0.625	< 0.01
	Interaction	0.659	-0.276	0.668	0.662	< 0.01

either published or found after this experiment was performed.

For Li et al. [61] their work was published at the same time as the experiment in this chapter, but are included in chapters 5 and 6.

Cavalin et al. [27] were found after all experiments were complete in this thesis, and have thus not been included as a baseline.

4.7 Results

4.7.1 Individual Feature Results

In this section we report our results for each individual feature and classifier. We report F1, difference in F1 (F1 Δ) to our baseline (Eugenio), precision, accuracy, and its *P*-value in respect to our chosen baseline.

Table 4.8 shows our results for the event Death of a Parent. Content feature perform about the same as our baseline features. Semantic and Interaction feature have relatively

Table 4.9: Falling in Love Single Feature Classifier Results

Classifier	Feature Set	F1	F1 Δ	P	R	*P*-value
NaiveBayes	Eugenio	0.805	0	0.805	0.805	
	Content	0.809	0.004	0.809	0.809	0.628
	Semantic	0.597	-0.209	0.666	0.623	< 0.01
	Interaction	0.542	-0.263	0.58	0.564	< 0.01
J48	Eugenio	0.824	0	0.829	0.825	
	Content	0.81	-0.014	0.814	0.811	0.121
	Semantic	0.431	-0.393	0.498	0.498	< 0.01
	Interaction	0.548	-0.276	0.715	0.605	< 0.01
RandomForest	Eugenio	0.833	0	0.838	0.833	
	Content	0.827	-0.006	0.829	0.827	0.38
	Semantic	0.554	-0.279	0.598	0.577	< 0.01
	Interaction	0.498	-0.335	0.498	0.498	<0.01

poor F1 scores. However for interaction features, we see a good precision score when using a J48 classifier.

Table 4.9 shows our results for Falling in Love. We see content features and Eugenio again performing at a similar level, while semantic and interaction features under perform. Similar to Death of a Parent, for interaction features we again see a good precision score when using a J48 classifier.

Table 4.10 shows our single feature results for Getting Married. Content and Eugenio again perform very similarly, while semantic and interaction features under-perform in comparison. While not as good as our previous two events, Interaction features again has slightly higher precision than recall when using a tree based classifier.

Table 4.11 shows our results for the event Having Children. We again see Content features performing the same as our baseline, where interaction and semantic features under-perform. Random Forests appear to work the best across our three classifiers. Again, we see Interaction features showing higher precision than recall amongst all three classifiers,

Table 4.10: Getting Married Single Feature Classifier Results

Classifier	Feature Set	F1	F1 Δ	P	R	*P*-value
NaiveBayes	Eugenio	0.898	0	0.898	0.898	
	Content	0.905	0.007	0.905	0.905	< 0.01
	Semantic	0.638	-0.26	0.689	0.654	< 0.01
	Interaction	0.39	-0.508	0.541	0.509	< 0.01
J48	Eugenio	0.908	0	0.909	0.908	
	Content	0.903	-0.005	0.903	0.903	0.019
	Semantic	0.665	-0.244	0.685	0.671	< 0.01
	Interaction	0.601	-0.307	0.659	0.623	< 0.01
RandomForest	Eugenio	0.905	0	0.905	0.905	
	Content	0.912	0.007	0.913	0.912	0.011
	Semantic	0.664	-0.241	0.684	0.67	< 0.01
	Interaction	0.618	-0.287	0.626	0.621	< 0.01

Table 4.11: Having Children Single Feature Classifier Results

Classifier	Feature Set	F1	F1 Δ	P	R	p-val
NaiveBayes	Eugenio	0.92	0	0.923	0.92	
	Content	0.918	-0.002	0.918	0.918	0.213
	Interaction	0.479	-0.441	0.572	0.539	< 0.01
	Semantic	0.566	-0.354	0.602	0.584	< 0.01
J48	Eugenio	0.915	0	0.915	0.915	
	Content	0.912	-0.003	0.913	0.912	0.206
	Interaction	0.593	-0.322	0.67	0.621	< 0.01
	Semantic	0.591	-0.324	0.591	0.591	< 0.01
RandomForest	Eugenio	0.921	0	0.922	0.921	
	Content	0.922	0.001	0.922	0.922	0.81
	Interaction	0.645	-0.276	0.654	0.648	< 0.01
	Semantic	0.618	-0.303	0.618	0.618	< 0.01

Table 4.12: Starting School Single Feature Classifier Results

Classifier	Feature Set	F1	F1 Δ	P	R	p-val
NaiveBayes	Eugenio	0.919	0	0.921	0.919	0
	Content	0.917	-0.002	0.919	0.917	0.103
	Interaction	0.473	-0.445	0.528	0.518	< 0.01
	Semantic	0.635	-0.284	0.657	0.643	< 0.01
J48	Eugenio	0.928	0	0.933	0.928	0
	Content	0.931	0.003	0.936	0.931	0.019
	Interaction	0.598	-0.33	0.686	0.629	< 0.01
	Semantic	0.617	-0.311	0.648	0.629	< 0.01
RandomForest	Eugenio	0.932	0	0.938	0.932	0
	Content	0.932	0	0.934	0.932	0.928
	Interaction	0.595	-0.337	0.603	0.599	< 0.01
	Semantic	0.636	-0.296	0.663	0.646	< 0.01

especially Naive Bayes and J48.

Table 4.12 shows our results for Starting School. Again, we see content features and Eugenio performing about the same. Semantic and interaction features again both underperform in comparison, but we again see interaction features favouring precision over recall for all three classifiers. Random Forests appear to offer the best results again across our feature sets.

4.7.2 Best Combination Results

Table 4.13 shows the results of our best performing feature combinations against our baseline. For each event, we select the best performing baseline classifier to compare against. Our feature sets are represented as Content Features (C), Eugenio (E), Interaction (I), and Semantic (S). We report F1, difference in F1 (F1 Δ) to our baseline (Eugenio), precision, accuracy, and *P*-value.

As we can see, in all cases we see a positive improvement over our baseline (mean of

Table 4.13: Best Performing Combined Classifier Results

Theme	Feature Set	Classifier	F1	F1 Δ	P	R	*P*-value
Death of a Parent	E	RF	0.935	0	0.935	0.935	
	All features	RF	0.938	0.003	0.938	0.938	< 0.01
Falling in Love	E	RF	0.833	0	0.838	0.833	
	All features	RF	0.844	0.011	0.85	0.845	0.215
Getting Married	E	J48	0.908	0	0.909	0.908	
	All features	RF	0.916	0.008	0.917	0.916	0.011
Having Children	E	RF	0.921	0	0.922	0.921	
	All features	RF	0.923	0.002	0.924	0.923	0.489
Starting School	E	RF	0.932	0	0.938	0.932	
	All features	RF	0.937	0.005	0.941	0.937	0.01

+0.009 improvement), with Death of a Parent, Falling in Love, Getting Married, and Starting School suggesting a compatibility with Hypothesis 1. Random Forests appear to be the best performing classifier algorithm. In all cases, we see a combination of Content Features and our Eugenio baseline appear in each feature combination, with semantics appearing in three, and Interactions appearing in two.

4.7.3 Feature Analysis

In this section we analyse some of our top features extracted across each feature set. We measure the usefulness of a feature by its information gain.

4.7.3.1 Content Features

Table 4.14 shows our top five features for each event for Content Features. Quoted features represent n-grams, while non quoted represent metric based feature sets.

As we can see, most of our content features have a very high information gain, although a number of these features correlate to some of our keywords we used to collect the dataset

Table 4.14: Content Feature Analysis

Event Type	Feature Name	Information Gain
Death of a Parent	"pass"	0.258
	"awai"	0.23
	"pass awai"	0.218
	"father"	0.129
	time of day	0.115
Falling in Love	"love"	0.3974
	polarity	0.1557
	time of day	0.1057
	"happi"	0.0817
	"school"	0.0453
Getting Married	"knot"	0.1528
	time of day	0.1098
	"marri"	0.105
	"wed"	0.1035
	"ti"	0.0642
Having Children	"birth"	0.189
	"deliv"	0.1488
	"babi"	0.1433
	time of day	0.1274
	"child"	0.0632
Starting School	"school"	0.5809
	"start"	0.3159
	informativeness	0.1075
	time of day	0.0927
	"start school"	0.0841

Table 4.15: # of semantic features extracted

Feature Type	DoP	FiL	GM	HC	SS	Average
# Entities	23	6	67	28	17	*28.2*
# Concepts	52	22	73	53	46	*49.2*
Total	75	28	140	81	63	*77.4*

in section 4.3.1.

We also see only two bi-grams above uni-grams appearing in our top results. This backs up what Eugenio et al [43] saw in their work with the addition of bi-grams and tri-grams offering little improvement over a more basic bag of words model.

We do see a number of our metric based content features appearing in our top features. Time of day, polarity of post, and informativeness all have high information gain scores. Time of day appears to affect every event, suggesting a pattern of when certain types of events are posted on Twitter.

Polarity appears highly in Falling in Love. Considering the top n-gram for that event is "love", this may explain the high ranking of polarity as love is possibly a high scoring sentiment lexicon.

Informativeness appears in Starting School, suggesting the vocabulary used in those posts is more unique than that of the rest of Twitter.

4.7.3.2 Semantic Features

Table 4.15 shows the total number of concepts and entities extracted per event, while table 4.16 shows the top five semantic features per theme.

As we can see, in most cases our top features do not have very high information gain scores. Interestingly, we see the concept of URL appear in three of our themes, suggesting

that there may be a correlation between hyperlinks and particular events occurring.

4.7.3.3 Interaction Features

Table 4.17 shows the information gain for each of our interaction features against each event. As we can see, conversation length, the probability a tweet is favourited, and the probability a tweet is retweeted, all have relatively high information gain scores. We also see the mean number of user interactions on a tweet has some predictive power, but the number of favourites, number of retweets, and the time a conversation goes on for, does not appear to correlate against our class distributions.

4.8 Discussion

In this chapter we have demonstrated a number of different feature sets that have expanded on the state of the art, and have shown that with relatively simple techniques, we can improve upon previous results.

When comparing content features against our baseline, we agree with Eugenio that the addition of bi-grams, and tri-grams offers little improvement over using just uni-grams. However, this could be partly down to how our datasets are extracted. After ranking top content features by information gain, we found a number of keywords that were used to collect our datasets with (Section 4.14).

Considering how many keywords were in our top features, there is a chance that weaker, but useful, bi-grams, or tri-grams are not utilised by our classifiers, instead favouring the strongly performing keyword uni-grams.

The easiest solution to this would be to mask the keyword in our training sets for each

Table 4.16: Top Semantic Features per Event

Event Type	Type	Feature Name	Information Gain
Death of a Parent	Concept	URL	0.0166
	Concept	Duration	0.01582
	Concept	Person	0.012
	Entity	http://en.wikipedia.org/wiki/Cance	0.011
	Concept	Work	0.010
Falling in Love	Entity	http://en.wikipedia.org/wiki/Girlfriend	0.0314
	Concept	Time	0.0194
	Concept	Country	0.0178
	Entity	http://en.wikipedia.org/wiki/Birthday	0.0178
	Entity	http://en.wikipedia.org/wiki/Knot	0.0178
Getting Married	Entity	http://en.wikipedia.org/wiki/Knot	0.0835
	Concept	Person	0.0543
	Concept	URL	0.0501
	Concept	Agent	0.0309
	Concept	Actor	0.0255
Having Children	Concept	URL	0.01911
	Concept	Disease	0.00951
	Entity	http://en.wikipedia.org/wiki/Shakira	0.00926
	Entity	http://en.wikipedia.org/wiki/Etsy	0.00926
	Concept	Duration	0.00777
Starting School	Concept	Time	0.0468
	Concept	Person	0.0232
	Concept	Agent	0.023
	Concept	Artist	0.0193
	Concept	Work	0.017

Table 4.17: Interaction Feature Information Gain

Feature	DoP	FiL	GM	HC	SS	Average
Conversation Length	0.1522	0.135	0.1605	0.1477	0.1512	*0.14932*
P (Favourite)	0.1609	0.011	0.1564	0.1477	0.1512	*0.12544*
P (Retweeted)	0.1110	0.106	0.1247	0.1187	0.0975	*0.11158*
Mean User Interactions	0.0174	0	0.0153	0.0296	0.0285	*0.01816*
# Favourites	0	0	0	0	0.0155	*0.00310*
Conversation Time	0	0	0	0	0	0
# Retweets	0	0	0	0	0	*0*

tweet. However, this itself is problematic due to how short a tweets text is. For example, take the following tweet from our dataset:

At my sisters wedding!

In this example, *wedding* was the keyword that gathered this tweet, but when we remove it, the tweet changes context and has little to no relevance to the event.

At my sisters!

In addition, removing words form our tweets decreases the chance of finding common bi-grams, and tri-grams.

Because of this, rather than mask out our keywords, we propose a new dataset collection strategy later on in section 6.2. The method proposed should allow us to remove collection keywords with significantly less loss of content.

Besides n-grams, we also saw several content metrics prove to be useful predictors. *Time of day* appeared in all five of our events, suggesting that certain events are typically posted at specific times of day. We also see the *polarity* of a tweets sentiment effect falling in love, which makes sense considering the expected vocabulary of tweets for that event are going to have very positive lexicon scores. *Informativeness* also appeared in Starting Schools top features, suggesting certain unique terminology is used in tweets about that particular event.

When we look at the performance of our semantic features we see that while by themselves they are weaker predictors than content or uni-grams, combined with other features sets they can help improve performance. Our best performing semantic features typically seemed to be more generic categories such as *Person*, *Work*, and *Agent*. We believe this overlap in common semantic relations can be expanded upon, which we investigate in Chapter 5.

Interaction features too have been shown to have a predictive power over certain datasets. With only seven features, we did not expect them to outperform other feature sets, but for two of our events, Having Children and Death of a Parent, we saw interactions out perform semantics. Tree based strategies also worked particularly well with interaction features, and appeared to bias towards a higher precision than recall, which considering our task is more beneficial. Given some of our motivations in section 1.1, a higher precision would be far more useful than a high recall. For example, in the case of managing our social media data, the objective is to reduce the number of posts to a subset of ones that are more salient. If most events are returned, but with a large number of false positives, the usefulness of the system is muted.

As interaction features only consisted of seven features, four of which were categorised as useful (section 4.7.3.3), there is a high chance of under-fitting on most models with them. Given our results, it looks as if tweets about events typically have different conversation lengths, and a probability score correlating to a tweets number of favourites and number of re-tweets. We also see a slight bias in the mean number of user interactions on tweets about most life events (except Falling in Love). We also see interaction features contributing to significantly better classifiers than our baselines for two of our events, Death of a Parent and Falling in Love.

In this discussion, we have already made a quick mention of flaws in our dataset collection strategy, but there also lessons learned with respect to our annotation strategy as well.

First, the subjective nature of answering whether some of the tweets were about, or not about, a particular event may have caused issues. For example, Falling in Love is by common sense, a very difficult life event to represent in a tweet. The removal of these types of

subjective events might be best considered, and replaced with easily identifiable ones.

Secondly, we see issues raising from the inclusion of non personal twitter accounts. For example, for our theme Getting Married, we see a number of highly performing n-grams such as *ring bearer pillow* and *http://en.wikipedia.org/wiki/Etsy*. Looking through our twitter dataset shows a number of tweets that were collected about adverts. Avoiding these types of tweets is not possible with the limitations of our dataset collection strategy, but we could ask annotators to label tweets with such a discrimination.

Finally, we must consider that Twitter itself is possibly not the ideal discourse platform for sharing personal life events. As already mentioned in section 1.2 other sites like Facebook and Instagram, might be more suitable for extracting these types of events from. As Instagram is already one of our main research questions in section 1.3, we address this later on in Chapter 6.

There were also also issues with our labelling strategy for our classifier. Our approach only considered including those tweets that were related to the theme, and to balance our classes and attempt to avoid bias, we sampled our negative class independently from theme.

There are two issues with this approach. First, even though we sampled our negative classes independent of theme to avoid bias, we still found significant patterns that correlated heavily towards other themes. For example, for our Getting Married classifier, in the real world we would not expect *pass awai* (picked from sampling negative classes from Death of a Parent) to be a strong feature indicating that a tweet was not about an event.

Secondly, this approach does not take into account expected noise from Twitter, with posts that were not collected using our query words from table 4.1. Thus we need to consider filling our negative class with real world *random* tweets to create a more generalised

classifier.

In terms of our classification strategy we compared three types of classifiers: probability(Naive Bayes), tree (J48) and a tree ensemble (Random Forest). We found that for most cases, Random Forests performed the best, with J48 performing just slightly better than Naive Bayes. Future work should probably look at other types of classifiers like SVMs. In addition, considering the number of features extracted with techniques such as uni-grams and content features, we may find that there are a number of redundancies within our feature sets. Feature reduction techniques like ranking by information gain, of CFS Subset[49] selection might help improve accuracy for some of our feature sets.

4.9 Limitations and Recommendations

In this section, we highlight any limitations with the methodology within this chapter, as well as recommendations from this chapter.

4.9.1 Limitations

While we have shown good results for classification performance in section 4.7, these should be viewed in the context of some limitations with the methodology.

Firstly, our dataset collection strategy in section 4.3.1 used a keyword method to search for related tweets which were not removed as part of feature extraction. This in turn will have added bias to the classification performance as each tweet will have always contained one of the keywords. In addition, each tweet in our negative class will also have contained at least one of our collection keywords. As already discussed in the previous section (4.8), this appears to have boosted the usefulness of certain features which you would not expect

in a real world sample.

Related to this is another limitation on the research, which is the size of our datasets. Due to funding limitations we could only annotate 16k tweets, where from this dataset only 2241 tweets were in a positive class. A larger dataset may have elicited more interesting results, and shown larger differences across methodologies, including our chosen baselines.

4.9.2 Recommendations

With the limitations in mind from the previous section, we can provide several recommendations from the work done in this chapter.

First, we have shown it is possible to produce Twitter specific life event classifiers for Getting Married, Having Children, Death of a Parent, Falling in Love and Starting School. The best performing classifiers have all been a combination of features within this chapter: Content, Semantic and Interactions, using a RandomForest classifier.

It is also worth noting that while interaction features by themselves did not perform well, as discussed in section 4.7, there were three useful features: Conversation Length and the probabilities of a tweet being favourited or liked n times.

4.10 Summary

In this chapter, we have collected a new gold standard dataset and demonstrated several different feature sets against the current state of the art in personal event detection on social media. We have shown that a combination of new feature sets, including content features, semantic and interactions, suggests an improvement over our baseline.

5

Frequent Sub-Graph Mining of Syntactic and Semantic Graphs

In the previous chapter we started our investigation into detecting personal life events on Twitter by creating classifiers that used Content, Interaction, and Semantic features. In this chapter we look at investigating a different approach, by expanding tweets into syntactic and semantic feature graphs, and then mining frequent sub-graphs to be used as features in our classifiers.

5.1 Introduction

So far, our research has looked into extracting personal life events from social media by generating a new dataset, and using content, semantic, and interaction features to identify personal life events. We found that while content features performed slightly better than unigrams, there was only a small jump in performance. In addition, there were issues with our dataset composition for our classifiers. When building our negative class with tweets that were thematically independent, we found strong features that were unlikely to perform well outside our dataset.

Also, considering our Twitter dataset, the problem of identifying personal events in tweets is slightly more nuanced than saying it is about an event or not. We found many tweets in our annotated dataset that while were thematically similar, were not in fact about a personal event. For example, we found a number of tweets that were about adverts to do with buying wedding rings, or competitions for Gary Barlow to sing at weddings.

Finally, given the poor performance of semantics in the previous chapter, we can consider ways to extend the semantic meanings within each tweet to attempt to capture more general patterns.

To this end, this chapter deals with answering the following research question:

R.Q. 2: *- Could we improve classification performance by extending the syntactic and semantic context of a tweet using graphs?*

To answer this question, and address some of our previous discussion points, we start this chapter by introducing the concept of frequent graph pattern mining, and define several types of syntactic and semantic graphs that we can generate from our tweets. We then

introduce a randomly sampled dataset collected from Twitters sample endpoint, and add a third label to our classifier, *About Theme*. We also introduce a second baseline to compare our results against the current state of the art.

We show that by including frequent sub-graph patterns as features in our classifier, we can improve F-Measure between 0.02 and 0.04. In addition, we also observe a distinction between the roles our semantic and syntactic patterns play: the former dealing with theme separation, and the latter with event.

To summarise:

- Introduce a semantic and syntactic graph-based approach for identifying personal life events from social media posts
- Add a third label to our classifier
- Enhance our current dataset with a randomly sampled twitter dataset
- Mine frequent graph patterns and serialise them as features in a LibLINEAR classification algorithm
- Compare results to state of the art, showing an average increase of 0.03 in F1
- Discuss the advantages of using syntactic and semantic graphs for personal life event classification.

The rest of this chapter is structured as follows. In Section 5.2 we present an overview of our system, including a pipeline to extract *Feature Graphs* from our datasets, and how we then use this in our classifiers. After that, in Section 5.3 we introduce our different types of features graphs, followed by Section 5.3 where we describe how each type of feature graph is generated. Section 5.4 introduces frequent graph mining, and how we implement it in our pipeline. Then we introduce our evaluation set-up in Section 5.5, followed by our results in

Section 5.6. Finally, we present a discussion of our results in Section 5.7, with a summary of this chapter in Section 5.9.

This chapters work is supported by Dickinson et al. [41], and Saif et al. [98], as described in appendix Section 1.5.

5.2 Graph Mining for Event Detection

As previously mentioned in Section 4.8, we saw semantic features performing poorly compared against both our baseline, and extended content features. One issue behind this might be because of a lack of overlapping concepts and entities. For example, one post might mention the concept of a "Mother", while another mentions that of a "Father". When considering both of these terms, we understand that they refer to parents, but from our classifiers point of view, they are two distinctly separate concepts.

In addition to this, when considering either uni-grams or *n*-grams, we only consider phrases. There may be use in considering the syntactic structure of tweets instead, finding not just a common series of tokens, but looking at their POS, as well as how tokens connect in other grammatical styles.

Our proposed solution to this is to consider expanding each post into various graphs, semantic, and syntactic, then across our training set, look at mining frequently co-occurring sub-graphs. To our knowledge, this strategy has not been employed within event detection, although has been used in classification in fields such as cheminformatics (Huan et al. [51] Deshpande et al. [37] Bandyopadhyay et al. [15]).

To this end, we first introduce two types of graph categories, syntactic (5.2.1) and semantic (5.2.2). We then present a pipeline for extracting frequent sub-graphs to be used as fea-

tures in our classifiers (5.2.3).

5.2.1 Syntactic Graphs

Syntactic graphs are the first type of graph we would like to introduce. We consider syntactic graphs to deal with the syntactical relationships between both tokens, and part of speech, within a tweet.

There are two types of syntactic graphs that we look to extract: Token, and Dependency. Token graphs deal with the standard ordering of words, and can be considered as related to *n*-grams, but also consider the part-of-speech relationships between tokens as well. Dependency on the other hand, looks at the dependency grammar[80] within tweets, and models that as a graph.

5.2.1.1 Tokens

A token graph models token ordering between both tokens and POS tags. It models each token and POS as a node, and a relationship indicating which node comes next in the sentence. For example, consider the text:

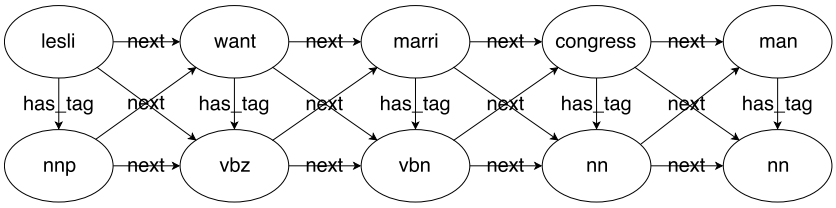
Leslie just wants to be married to a congress man

Figure 5.1 shows this example represented as a token feature graph, after stemming and stop word removal.

A viable sub-graph in this example could be (vbz)-[next]->(marri), indicating a verb followed by the token marri.

Representing the graph this way then gives us three types of features that can be extracted:

Figure 5.1: Token Graph: *Leslie just wants to be married to a congress man*



Token Order Similar to *n*-grams, we can look to extract common pairings of tokens. However, rather than restrict ourselves to uni-grams, bi-grams, and tri-grams, we can extract all lengths of *n*-grams within a tweets sentence (that occur more than a specified minimum support).

POS Order Rather than just considering *n*-grams, we can also consider the ordering of each tokens POS. There may exist syntactic POS patterns specific to each type of event that prove to be useful features.

Token POS Patterns We can also look at common patterns where we link tokens and POS together. For example, we may find patterns such as (NNP)-[next]->(marri), which can cover a wide range of examples. E.g., 'Lesli marri', 'Mary marri', 'Chuck marri', are all covered by this pattern.

5.2.1.2 Dependency

Our second syntactic type of graph we want to extract is the dependency grammar within a tweet. Dependency grammar considers a richer form of relationships between tokens, which is already represented within a graph format. Between two tokens, there exists a governor (also referred to as a head or regent) and a dependent (also refereed to as modifier),

and some relationship between them. Governor tokens themselves can also be dependent of another token, although in each sentence, there exists one governor token which is not a dependent itself and is considered the root of the sentence. For this type of extraction, we have decided to use the StanfordNLP Neural Network Parser[28]. There exist other types of dependency parsers (e.g.[56]), however due to technical reasons, we have decided to use the Stanford NLP parser as it integrates and performs well with our pipeline.

Take the example tweet:

My Father passed away

Figure 5.2 shows this example as a dependency feature graph. For visualisation purposes, we only show the tokens, however, the same concept of interchanging tokens and POS as we do for token graphs in Section 5.2.1.1 is applied in our work.

Similar to token graphs, there are three types of features that can be extracted:

Dependency Tokens: We can extract common dependency orders and relationships, mined directly from our graphs. Figure 5.3 shows an example sub-graph. This can be considered similar to *n*-grams, but using instead dependency grammar rather than token order as relationships.

Dependency POS: Rather than just mining token values, we can also mine their POS to find dependency syntactical patterns. Figure 5.4 shows an example sub-graph.

Dependency Token POS Patterns: Finally, we can look at interchangeable patterns between our tokens and POS. Figure 5.5 shows an example sub-graph.

Figure 5.2: Dependency Graph: *My Father passed away*

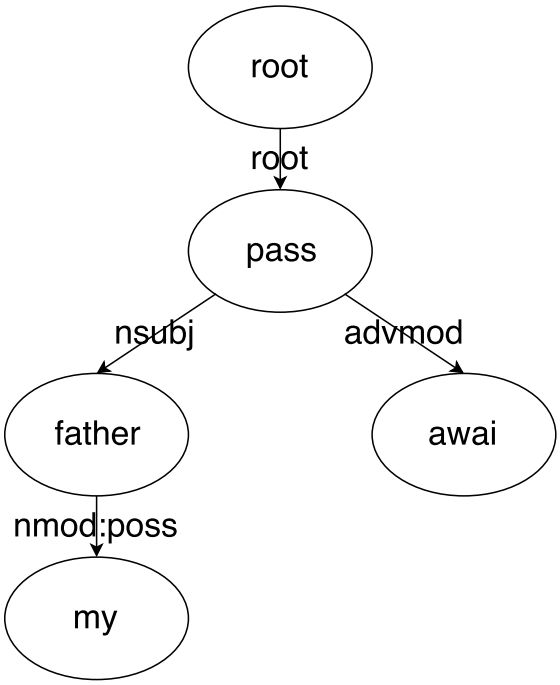


Figure 5.3: Token Dependency Subgraph Example

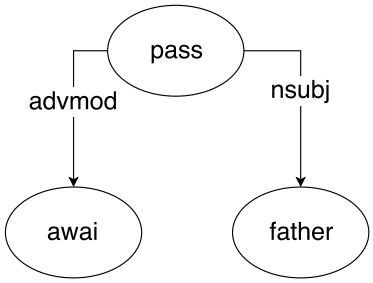


Figure 5.4: POS Dependency Sub-graph Example

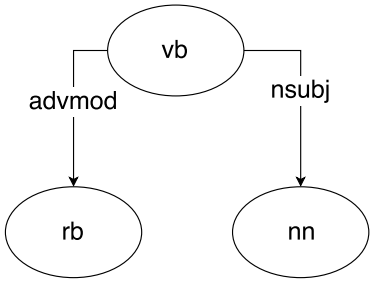
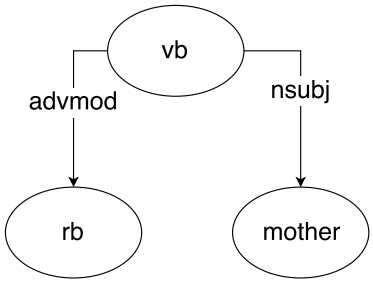


Figure 5.5: Token & POS Subgraph Example



5.2.2 Semantic Graphs

In Chapter 4, we showed that while semantic features do contribute to classifier performance, as a feature by themselves they were not particularly strong. As an alternative, we suggest generating semantic graphs from two popular networks WordNet, and ConceptNet, with the aim of finding common semantic sub-graphs between our posts.

Our hypothesis for this is that by expanding posts into semantic networks, we find more overlap in concepts, rather than detecting what concepts exist within each tweet.

The intuition behind this hypothesis is best served with an example. Given one of our events, e.g., *Death of a Parent*, let us assume we detect the concept *mother n* times, and *father m* times. When comparing such patterns, they are considered distinct and offer no relation to each other. However, if we expanded each $n + m$ posts into a semantic network that linked both to the concept of a *parent*, *parent* would occur at least $n + m$ times as it then turns into a shared concept.

While there exist many types of networks that we could potentially expand into, our starting points include WordNet, and ConceptNet.

5.2.2.1 WordNet

WordNet is a large lexical database of English, with nouns, verbs, and adjectives all grouped together as synonyms (referred to as synsets).

Our proposed method, is to extract synsets from tweets, and expand them using hypernym, and hyponym relationships into WordNet.

Hypernyms and hyponyms represent an *is a* relationship, where hypernyms can be considered a superclass of a hyponym, or a hyponym can be considered a subclass of a hypernym. For example, *colour* is a hypernym of *purple*, as colour is the more generic parent of purple, whereas *violet* is a hyponym of *purple* as it is a specific shade of the broader colour.

Figure 5.6 shows an example of a tweet expanded into WordNet at a depth of 1 for both hypernyms and hyponyms.

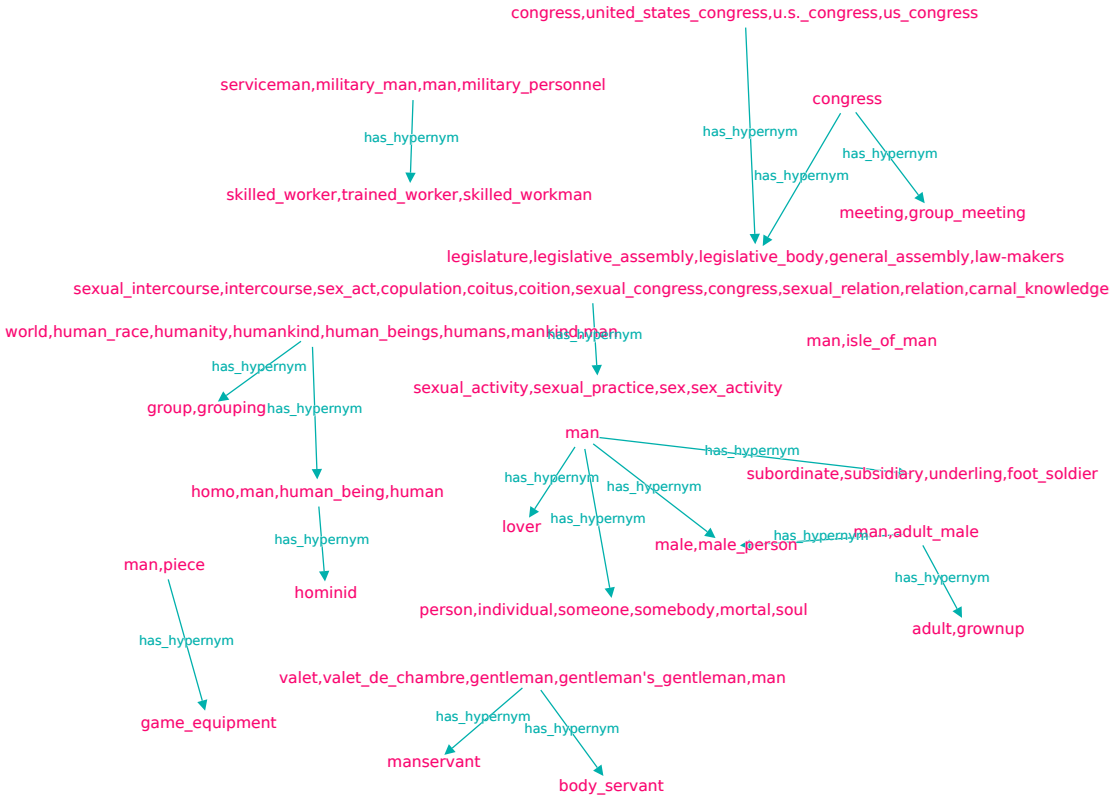
Our hypothesis for this is by expanding into WordNet, we may find frequent patterns shared within a set of tweet posts that are useful features. As we expand into WordNet, there is a greater chance of crossover between synsets, although only up to a point as the more generic our synsets become, the more noise we are likely to introduce into our feature set.

5.2.2.2 ConceptNet

ConceptNet is a hypergraph of relationships between concepts. Originally it was based on the Open Mind Common Sense[107] project, although more recently has included several other knowledge sources:

- OMCS English, Portuguese, Japanese, Dutch, Korean, French (ConceptNet 4)
- OMCS Chinese

Figure 5.6: WordNet Graph: *Leslie just wants to be married to a congress man*



- GlobalMind
- Verbosity
- Wiktionary translations
- WordNet
- DBPedia's type and location relationships
- JMDict
- OpenCyc via Umbel

Our hypothesis with ConceptNet is the same as WordNet, where we look at extracting concepts from tweets (detailed in Section 5.3.3), and then expanding into ConceptNet. It is worth to point out that WordNet relationships exist within ConceptNet, however due to the sheer size of the graph, we are limited to how far we can expand into ConceptNet, before we see severe performance degradation with our sub-graph mining (detailed in Section 5.4.2.2). Thus including WordNet as a separate feature set, allows us to consider deeper expansions, than what we would be limited to with ConceptNet.

Figure 5.7 shows an example of our ConceptNet graph.

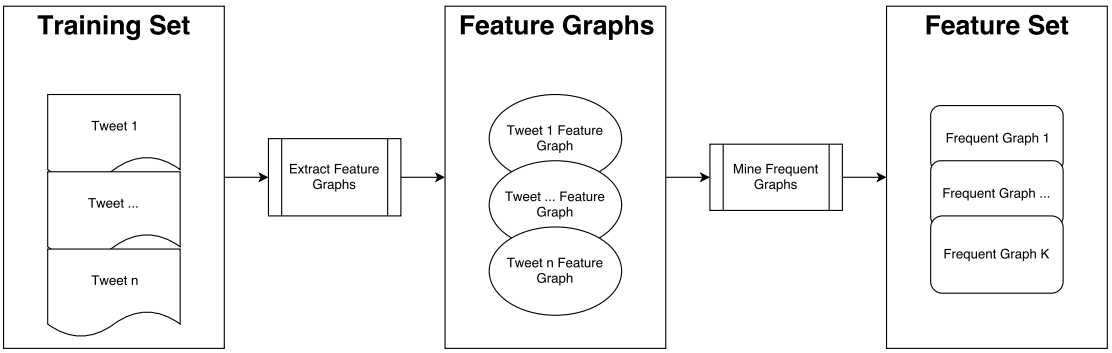
5.2.3 Pipeline

In order to convert our tweets into feature sets, we present in this section a pipeline as illustrated in Figure 5.8.

Let there exist n tweets in a given dataset. Our first step is to generate one of our four types of feature graph for each tweet in a one-to-one relationship, thus for n tweets we end up with n feature graphs (Section 5.3 explains how we extract each type of feature graph).

Figure 5.7: ConceptNet Graph: *Leslie just wants to be married to a congress man*

Figure 5.8: Frequent Graph Mining Pipeline



Given these *n* feature graphs, we then provide these as input to a frequent sub-graph mining algorithm, along with our chosen minimum support, to mine frequent sub-graphs.

The output of this process then generates a feature set for us to use in a classifier. Any new tweets to be classified then have to have their text converted into the corresponding feature graph, and every sub-graph checked against it to see if it exists or not. Our representation of this is a binary vector where we indicate a sub-graph exists by 1, and 0 if it does not.

5.3 Creating Feature Graphs

In the previous section we introduced each of our different feature graphs. In this section we go into more detail as to how we create each of our feature graphs. Section 5.3.1 deals with the generation of our syntactic graphs, while sections 5.3.2, and 5.3.3 deal with generating WordNet, and ConceptNet respectively.

5.3.1 Syntactic Graph Extraction

First we pre-process our token graph, removing stop words, and stemming tokens (as illustrated earlier on in Section 4.5.1). After this, we then treat each token as a node in our graph. There are two types of relationships: Token relationships, and dependency relationships. For token relationships, we simply add the edge *HAS_NEXT*, indicating the next token in the string. We also add POS tags as nodes to our graph, with a *HAS_TAG* edge from the respective token to POS node, as well as a *HAS_NEXT* tag between a token, and its next tokens POS node (as well as each POS node, and the next token node).

For dependency relationships, we use Stanford NLP DependencyParser class, which is an implementation of work done by Chen and Manning[28]. The output of the class is in a graph format already, so all we have to do is convert to a graph format that our sub-graph mining algorithm can accept.

5.3.2 WordNet Graph Extraction

To extract WordNet graphs, we tokenise each sentence using our preprocessing strategy from Section 4.5.1. For each non stop-word token extracted, we then use the JWI library[45] to search for synsets related to that word, using it and the corresponding POS tag. We do not perform any disambiguation between synsets, and simply add all found to the graph.

Once we have extracted all synsets from the tokens within the text, we then look at expanding our graph via hypernym and hyponym relationships, to a depth specified in the request (depth shown in Table 5.9).

5.3.3 ConceptNet Graph Extraction

To extract ConceptNet graphs, we follow the same pre-processing strategy as WordNet.

We use an off-line version of ConceptNet ran in Docker[52], and for every token, generate a candidate concept URI. ConceptNets URI format takes the form:

$$/c/en/concept\_name$$

The prefix /c/ indicates a concept, /en/ that it is english, then the name (with multiple words separated with _). To reduce complexity for this experiment we only consider single token concepts, although there is scope using concept detection such as[82] to extract multi worded concepts.

For each candidate concept URI, we then check ConceptNet to see if it exists. If the concept does, we then add the concept to our graph.

Given this node, we then expand into ConceptNet, adding all relationships from that Concept, until we reach a given depth (depth shown in Table 5.9).

5.4 Frequent Graph Mining

In this section, we describe in more detail how we mine our feature graphs for frequent sub-graphs, as well as several optimisations we make to our mining process.

5.4.1 Frequent Graph Mining Algorithms

To mine these graphs, we use the Parallel Sequential Mining Suite (ParSeMiS). This suite contains graph algorithms such as: gSpan[126], CloseGraph[127], Gaston[79], and Dagma[121].

Table 5.1: Frequent Mining Algorithm Run Times

Algorithm	# Frequent Patterns	Run Time (seconds)	Total Instances
gSpan	221059	27.9	1161
Gaston	355941	95.23	1161
CloseGraph	9767	5.69	1161

Table 5.1 shows each algorithm against our Having Children dataset for just token graphs, one of our smallest graph sizes, with a minimum absolute support of 2. We report the number of frequent patterns mined, and the total run time each algorithm took.

As we can see, both gSpan and Gaston extract more than 20 times as many patterns as CloseGraph. As each pattern would be represented as a feature within our classifier, there would be 200 times more features than instances, vastly increasing the risk of the curse of dimensionality[55]. In addition, the frequent sub-graphs that CloseGraph are considered closed and are a form of lossless pruning.

A closed frequent subgraph is defined as not existing within a supergraph with the same number of occurrences. For example, consider we extracted two single node frequent subgraphs *(A)* and *(B)* with respective supports 10 and 12.

In addition, we also extract a supergraph of the two *(A)-(B)* with support 10.

Only two of these graphs can be considered closed (*(B)*, and *(A)-(B)*) as neither have a frequent supergraph with the same support. However, *(A)* has the same support as the supergraph, meaning it only ever occurs when paired with B. In terms of using this as a feature set within our classifier, *A* can be considered redundant as it has the same information as (A)-(B) does.

We also see CloseGraph perform significantly faster than both gSpan, and Gaston. While the time difference reported here is not too large, with feature sets like ConceptNet, where

Table 5.2: WordNet Extraction

WordNet Depth	Nodes	Edges	Size of Graph (kb)
0	16	0	0.95
1	77	62	8.965
2	121	113	15.2845
3	156	146	19.66

the size of each graph is far larger than that of a syntactic graph, run time becomes important.

Because of these results, we have decided to opt for using CloseGraph to help both reduce the number of features in our datasets, as well as decrease the run-time for mining each of these features.

5.4.2 Semantic Depth

As stated in Section 5.1, our hypothesis is that the expansion of semantic networks can help identify common parent nodes that will help improve classification performance. However, the further we expand, the more noise and redundant patterns we risk adding to our training sets. In this section, we consider how far we can viably walk into both WordNet, and ConceptNet, when expanding our posts into semantic graphs.

5.4.2.1 WordNet Depth

When generating our WordNet feature graphs, we need to consider how far we expand into WordNet. Table 5.2 shows some metrics for WordNet extraction for Getting Married. We show the median number of nodes, edges, and size of graph. As we can see, WordNet graphs scale quite well, and moderately increase in size at each depth.

Figure 5.9: WordNet Depth Effect on F1

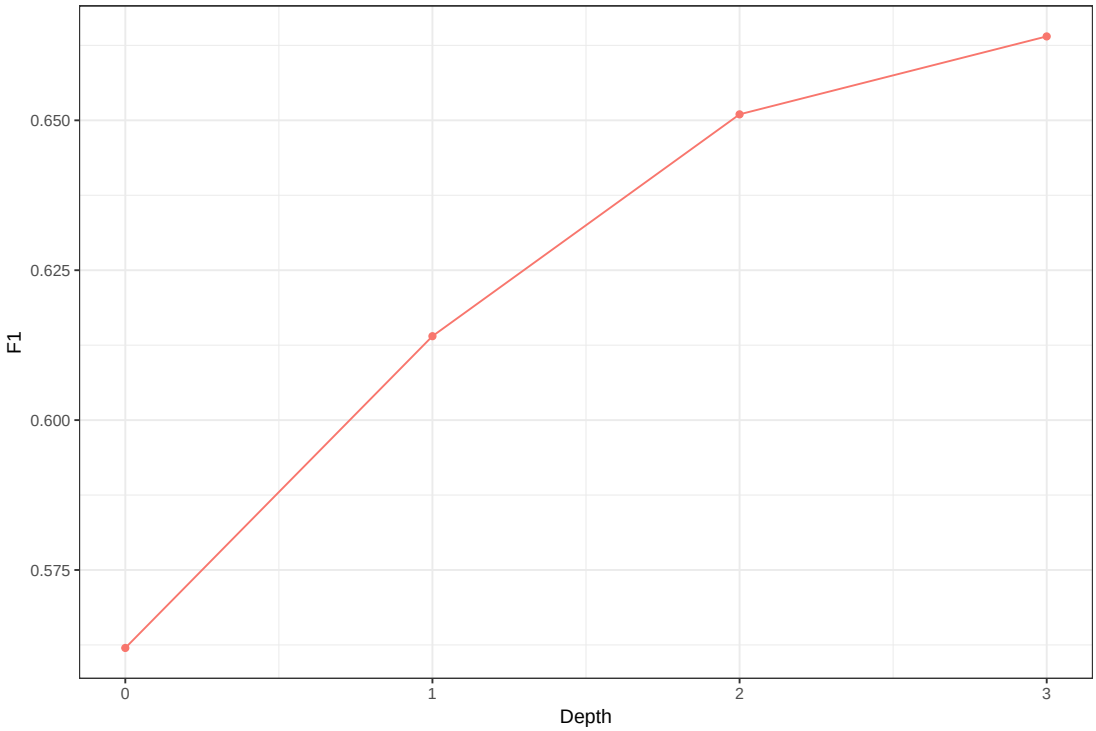


Figure 5.9 shows how F1 score is affected by depth for Getting Married. F1 is calculated with the same evaluation setup as outlined later in Section 5.5

5.4.2.2 ConceptNet Depth

Similar to WordNet, we need to specify a depth of how far we expand into ConceptNet. Unlike WordNet however, ConceptNet graph sizes expand at a much greater rate, hindering our performance on each expansion.

Table 5.3 shows some metrics for extracting ConceptNet Graphs for Getting Married. We show the average number of nodes, edges, size of graphs, and generation time.

As we can see, ConceptNet exponentially increases with depth the size of our graphs, as

Table 5.3: ConceptNet Extraction

ConceptNet Depth	Nodes	Edges	Size of Graph (kb)	Generation Time (s)
0	4.82	0	0.25	0.17
1	87.03	88.15	10.73	0.25
2	947.37	1185.05	137.26	0.83
3	8599.40	12238.07	1394.56	7.69

well as the time it takes to generate.

5.4.3 Minimum Support Selection

For each type of graph we want to extract, minimum support selection is crucial. Too high and we extract only generic graphs across our classes, too low and we extract too many features that can cause over-fitting. In this section, we look at the effect minimum support has on each of our graph feature sets, across each of our datasets, to allow us to select a suitable support value. We increase minimum support between 0.005, and 0.1, at steps of 0.005, for a total of twenty results per feature set. Due to the variations in F1 score over each event, we display the normalised F1 values.

Figure 5.10 shows our minimum support values for WordNet. As we can see from the graph, minimum support varies between each event. Having Children, Getting Married, and Starting School all benefit from lower minimum support values, while Death of a Parent, and Falling in Love peak in the middle of their range. However in all cases, we see F1 scores start to drop off the lower it becomes. A sensible minimum support value for all events seems to be around 0.03 to 0.035.

Figure 5.11 shows our minimum support values for ConceptNet. Unlike WordNet, we see more of a steady decrease as our minimum support values increase. Excluding big spikes,

Figure 5.10: WordNet Minimum Support

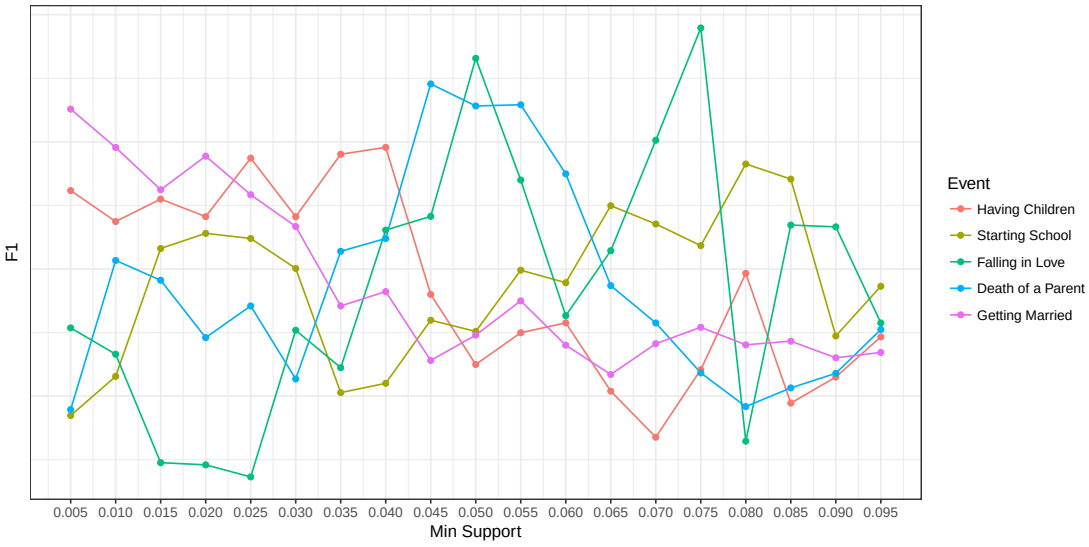


Figure 5.11: ConceptNet Minimum Support

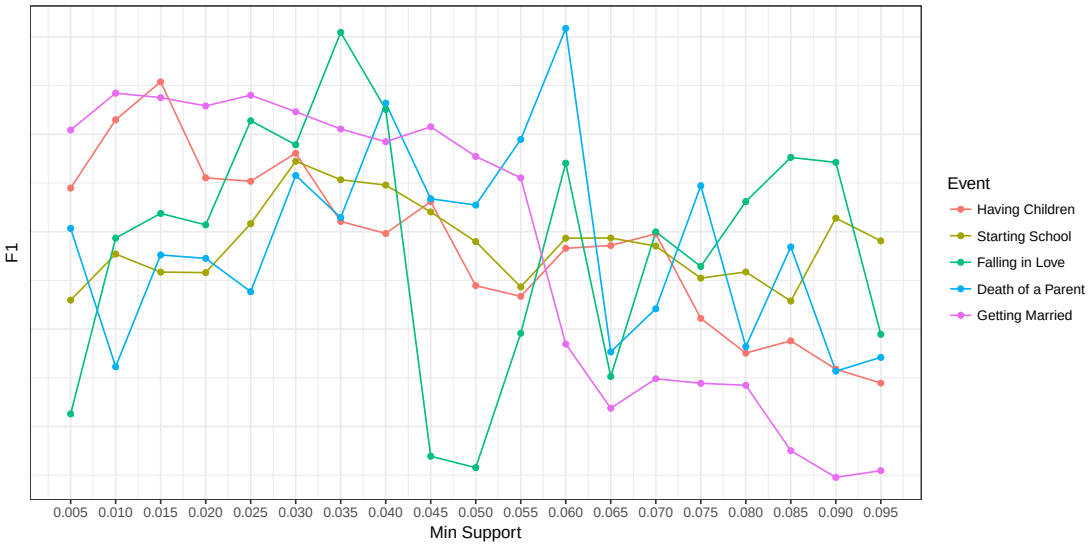


Figure 5.12: Syntactic Minimum Support

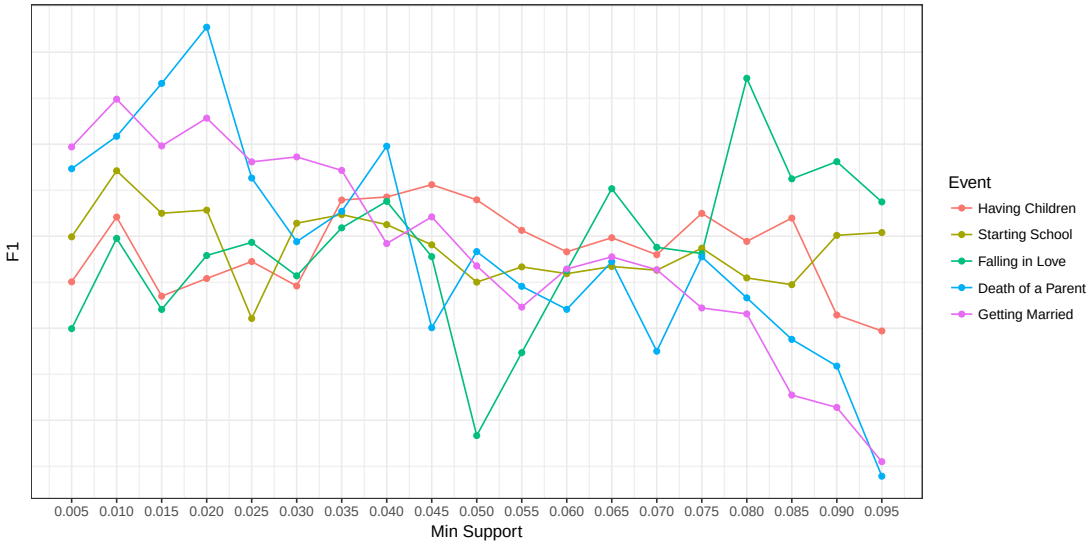


Table 5.4: Minimum Support Selections

Graph Type	Minimum Support
Syntactic	0.025
WordNet	0.03
ConceptNet	0.04

a sensible minimum support to set seems to be around 0.04.

Figure 5.12 shows our minimum support values for our Syntactic patterns. Again, we see a steady downward trend as we reduce our minimum support. A suitable minimum support to cover decent scores for most events seems to be around 0.025

To summarise, Table 5.4 shows our chosen minimum supports per graph type.

5.4.4 Including POS in Syntactic Graphs

In Section 5.2.1 we presented our syntactic graphs and the inclusion of POS within. However including POS could potentially lead to too many patterns that might cause over-

Table 5.5: POS in Syntactic Graph Results

Event	Without POS	With POS	Δ F1	P-value	# Instances
Death of a Parent	0.71	0.7	-0.01	>= 0.05	348
Falling in love	0.65	0.65	0	>= 0.05	342
Getting Married	0.73	0.75	0.02	< 0.05	2127
Having Children	0.71	0.74	0.03	< 0.05	1161
Starting School	0.68	0.71	0.03	< 0.05	1260

fitting within our classifier. In this section, we compare the inclusion, and exclusion of POS within our syntactic graphs.

To compare this, we extracted two types of syntactic graphs, those with POS included, and those without. We ran 10-fold cross validation 10 times using our LibLINEAR. Table 5.5 shows our results.

As we can see from the results, there is a clear split between when adding POS tags appear to improve results vs when it does not. For smaller sized datasets, we do not see a clear shift in F1 (-0.1 to 0), however for our larger datasets we see a positive F1 change between 0.02 and 0.03. This suggests that for our smaller training sets, there are not enough examples of when including POS might be useful. However with more instances, we see POS start to play a useful role in classifying posts.

Because of these results, when using syntactic features, we will include POS tags in our graphs.

5.4.5 Sub-graph Edge Limiting

One useful ability of ParSeMiS is to specify a maximum number of edges each sub-graph returns. For example, an edge limit of one would only return sub-graphs with no more than two connected nodes. As gSpan is a DFS algorithm that expands candidate frequent

Table 5.6: Trade off between edge limiting and F1

Edge Limit	Time Taken (seconds)	F1
No Limit	486	0.726
1	14	0.708

sub-graphs (2.3.1), smaller frequent subgraphs are found faster reducing both time and the amount of RAM to hold subgraphs in memory. While syntactic graphs are relatively small in size, we see memory constraints for our semantic graphs that need to be addressed.

Table 5.6 shows the trade off of only finding sub-graphs with one edge in terms of F-measure and time using the WordNet feature graphs from the Death of a Parent dataset. F1 is calculated with the same evaluation setup as outlined later in Section 5.5. As can be seen, there is a slight decrease in our F1 score by 0.018, however the time taken is reduced drastically from 8 minutes, to just 14 seconds.

Death of a Parent is one of our smaller datasets for this experiment, so this time difference is more important when considering datasets like Having Children, which can take several hours to mine our frequent graphs from. Thus for this experiment, we look to trade some of our F1 score in order to perform enough iterations of our experiment.

5.4.6 Mining Feature Sets Individually

While it is possible to mine our feature sets in one large graph, this is a very expensive operation. The number of nodes within a graph affects the performance of frequent graph mining libraries like gSpan, as it increases the number of permutations of candidate sub-graphs.

This in turn leads to large amounts of memory required to run our sub-graph mining algorithms. Table 5.7 shows the average graph size for Having Children, and each feature

Table 5.7: Average size of feature graphs

Feature Graph	# Nodes	# Edges	Size (kb)
All Features	457	539	65.485
Syntactic	28	118	9.137
WordNet	314	303	41.161
Concepts	118	118	14.801

graph type, as well as combined.

As we can see, when combined our graph sizes are much larger, and cause us memory issues when attempting to mine frequent sub-graphs from them.

As already mentioned in Section 5.4.5, for WordNet and ConceptNet, we can perform additional optimisations at the expense of some performance. However, this type of optimisation does not need to be performed on syntactic graphs due to their size, thus we can treat these differently when mining frequent sub-graphs from them.

5.4.7 Graph Pruning and Cycling Graphs

When expanding into semantic networks like ConceptNet, we may end up adding a number of redundancies to our graphs from simple expansion. For example, let us say we find the concept */c/en/baby* within our Having Children dataset and after expanding into ConceptNet, add a new node to our graph, such as */c/en/child* (e.g., *(/c/en/baby)-[IS_A]->(/c/en/child)*). If */c/en/child* only ever appears as an expansion of */c/en/baby*, this can be considered a redundant expansion.

In addition to this, we also look at generating our graphs as cyclic graphs. In terms of graph size in memory, this reduces repeated nodes, while still maintaining the structure of our graph.

Table 5.8: Graph size comparison with pruning

Pruned Methodology	Average Graph Size	Total Graph Size
Not Pruned	69kb	132mb
Not Pruned Cyclic	40kb	76mb
Pruned	35kb	67mb
Pruned Cyclic	18kb	34mb

Table 5.8 shows each methodology applied to our Getting Married dataset for ConceptNet features. We show the average graph file size in GML format[50], as well as the total serialised graph size for the dataset.

As we can see, converting graphs from acyclic to cyclic graphs reduce the size of each graph to about 42% of the original graph size. If we just prune our graphs, we see a reduction to 49.3% of the original graph size. When applying both pruning and cyclic graphs, we see the graph size reduce to 74.2% of its original size.

5.5 Evaluation Setup

5.5.1 Dataset

We use the same Twitter dataset we collected in Section 4.3, but with some modifications. One of the points we made in the discussion in Section 4.8 was how our dataset was balanced. We saw very high classification performance, but because we sampled our negative class from tweets across our five events, our negative label did not reflect a true representation of public tweets. Using such an unvaried dataset can cause bias in our negative classes, as we may end up discovering patterns to do with one of our five datasets.

To remedy this, we copy what Li et al. [61] do in their work, and create a negative class from tweets extracted from Twitters sample endpoint. These tweets are not annotated, but due

to our shared assumption with Li et al, the vast majority of tweets collected do not contain personal life events, and it is highly unlikely our random dataset contains enough false negatives to affect our results.

In addition to this, rather than focus on identifying whether a tweet is simply about an event, we try to distinguish whether it is just about a particular theme. In order to do this, we turn our classifier into a tri-class classifier with the following labels: +E+T (About Event, About Theme), (-E+T) (Not About Event, About Theme), -E-T (Not About Event or Theme).

5.5.2 Graph Feature Sets

In this section, we describe our graph feature sets to be used within our experiments.

As outlined in Section 5.4.6, each of our graph types are mined separately providing a separate feature vector for each type of graph. Therefore, we will first consider the performance of each of the three types: Syntactic, WordNet, and ConceptNet. Due to the number of features each one can produce, we will also compare the results with a feature selection technique outlined later in Section 5.5.4. The results for these experiments are in Section 5.6.1.

When comparing against our baselines, we will represent our graph feature as a single feature set called *graph*. This combined feature set will simply be the union of features mined from each individual graph. The results for these experiments are in Section 5.6.2.

Table 5.9: Graph Parameters

Graph Type	Parameter	Value
WordNet	Depth	2
	Remove Stopwords	True
	Minimum Support	0.03
	Maximum Edges	1
ConceptNet	Depth	1
	Remove Stopwords	True
	Minimum Support	0.04
	Maximum Edges	1
Syntactic	Remove Stopwords	True
	Stem	True
	Minimum Support	0.025

5.5.3 Graph Parameters

In Section 5.4 we performed several small experiments to consider what parameters we should choose for our mining task. Table 5.9 summaries our chosen values for our experiments.

5.5.4 Classifier Generation and Evaluation

For these experiments we use LibLINEAR[44] to compare our feature sets against. In our previous chapter, we considered several different algorithms, but found these can take a long time to tune correctly. As we are interested in the specific comparison of feature sets, we have elected to use a single algorithm instead.

To evaluate our results, we use WEKA's Experimenter[102], and perform 10-fold cross validation 10 times on our results. Each feature set is generated with the same instances, so we perform a two tailed paired student T-Test.

Due to the ratio of features to instances in some of our datasets, we also consider adding

a feature selection step. One easy method to reduce features, without tuning a parameter, is to set a threshold of 0 for some metric, i.e. removing all features with 0 variation. To this end, we have decided to run the same experiment above, and use a ranking feature selection strategy with Information Gain[116], excluding any feature with an information gain score of 0 or below. We perform this on every fold.

5.5.5 Baselines

For our baselines in this experiment, we re-use Eugenio[43] detailed in Section 4.6.4, as well as Content Features from Chapter 4.4.1. We also introduce a new baseline from Li et al[61], which utilises topic models.

The approach set out by Li et al is slightly more complex than Uni-grams and requires a bit more work and modifications to extract the required features. For starters, their approach is designed for a multi thematic classifier, based on topic models generated on dataset collection.

Their feature set is comprised of topic vocabulary words, windows around each word, and named entities extracted.

As we followed a different dataset creation method, we have had to adapt their approach slightly in reference to generating topic models. Our approach extracts $max(1, n - 1)$ topic models, where n is the number of labels in the classifier and must always be at least 1.

Table 5.10: Graph Results

Event	Graph Type	P	R	F1
Getting Married	ConceptNet	0.70	0.71	0.70
	Syntactic	0.74	0.75	0.74
	WordNet	0.65	0.66	0.65
Death of a Parent	ConceptNet	0.71	0.71	0.70
	Syntactic	0.71	0.69	0.69
	WordNet	0.74	0.73	0.72
Having Children	ConceptNet	0.67	0.68	0.67
	Syntactic	0.76	0.76	0.76
	WordNet	0.71	0.71	0.71
Falling in Love	ConceptNet	0.53	0.54	0.53
	Syntactic	0.65	0.65	0.64
	WordNet	0.54	0.54	0.53
Starting School	ConceptNet	0.65	0.66	0.65
	Syntactic	0.73	0.73	0.73
	WordNet	0.67	0.67	0.66

5.6 Results

5.6.1 Graph Results

In this section we present our results for each of our individual graph methods. We include these as Syntactic, WordNet, and ConceptNet. Table 5.10 shows each of our individual results.

We see syntactic results performing best in three of our events (Getting Married, Falling in Love, and Starting School), while WordNet appears to perform better for two (Having Children, and Death of a Parent). In general, we see ConceptNet perform at an average of 0.65. Syntactic at 0.71, and WordNet at 0.69.

Table 5.11 shows our results when we perform feature selection as outlined in Section 5.5.4

Table 5.11: Graph Results with Feature Selection

Event	Graph Type	P	R	F1	F1 Δ	*P*-value
Getting Married	ConceptNet	0.70	0.71	0.70	0.00	>= 0.05
	Syntactic	0.75	0.75	0.75	0.00	>= 0.05
	WordNet	0.66	0.66	0.65	0.00	>= 0.05
Death of a Parent	ConceptNet	0.71	0.70	0.70	-0.01	>= 0.05
	Syntactic	0.70	0.69	0.68	-0.01	>= 0.05
	WordNet	0.75	0.75	0.74	0.02	< 0.05
Having Children	ConceptNet	0.66	0.67	0.66	-0.01	>= 0.05
	Syntactic	0.77	0.77	0.76	0.01	>= 0.05
	WordNet	0.70	0.71	0.70	0.00	>= 0.05
Falling in Love	ConceptNet	0.61	0.61	0.60	0.07	< 0.05
	Syntactic	0.65	0.65	0.65	0.01	>= 0.05
	WordNet	0.58	0.58	0.56	0.03	< 0.05
Starting School	ConceptNet	0.66	0.66	0.65	0.00	>= 0.05
	Syntactic	0.73	0.72	0.72	-0.01	>= 0.05
	WordNet	0.68	0.67	0.64	-0.02	< 0.05

5.6.2 Baseline Results

In this section, we compare our graph approach against our baselines. Table 5.12 shows our results, where we report: F1, change in F1 relative to best baseline ($\Delta F1$), Precision(P), Recall (R), and the *P*-value relative to the best baseline.

For three of our events, Getting Married, Death of a Parent, and Having Children, we suggest a compatibility with hypothesis 2. However, for Starting School, we see no change in F1, and for Falling in Love, we see a decrease in performance of -0.03. This may possibly be due to size of our Falling in Love dataset, which with Death of a Parent, are significantly smaller than our others.

Thus to test this, as mentioned in Section 5.5.4, we also consider results where we perform feature selection. Table 5.13 shows our results for this.

Table 5.12: Graph Feature Baseline Results

Event	Feature set	P	R	F1	F1 Δ	*P*-value
Getting Married	Best Baseline: Content Feature	0.75	0.76	0.75		
	Eugenio et al. [43]	0.74	0.74	0.74		
	Li et al. [61]	0.73	0.74	0.73		
	Graph	0.76	0.77	0.76	0.01	0.011
Death of a Parent	Best Baseline: Li et al. [61]	0.75	0.74	0.73		
	Eugenio et al. [43]	0.74	0.73	0.72		
	Content	0.74	0.74	0.72		
	Graph	0.75	0.74	0.74	0.01	0.045
Having Children	Best Baseline: Eugenio et al. [43]	0.75	0.75	0.75		
	Li et al. [61]	0.7	0.7	0.69		
	Content	0.75	0.75	0.75		
	Graph	0.77	0.78	0.77	0.02	< 0.01
Falling in Love	Best baseline: Li et al. [61]	0.68	0.67	0.66		
	Eugenio et al. [43]	0.66	0.66	0.65		
	Content	0.65	0.66	0.65		
	Graph	0.64	0.63	0.63	-0.03	< 0.01
Starting School	Best Baseline: Content	0.73	0.73	0.72		
	Eugenio et al. [43]	0.72	0.72	0.71		
	Li et al. [61]	0.72	0.71	0.71		
	Graph	0.72	0.72	0.72	0	0.893

Table 5.13: Graph Feature Baseline Results with Feature Selection

Event	Feature set	P	R	F1	F1 Δ	*P*-value
Getting Married	Best Baseline: Content Feature	0.73	0.73	0.73		
	Eugenio et al. [43]	0.72	0.73	0.72		
	Li et al. [61]	0.72	0.73	0.72		
	Graph	0.77	0.78	0.77	0.04	< 0.01
Death of a Parent	Best Baseline: Li et al. [61]	0.76	0.74	0.72		
	Eugenio et al. [43]	0.74	0.71	0.71		
	Content Features	0.74	0.72	0.72		
	Graph	0.76	0.75	0.75	0.03	0.018
Having Children	Best Baseline: Eugenio et al. [43]	0.74	0.74	0.74		
	Li et al. [61]	0.71	0.71	0.7		
	Content Features	0.75	0.75	0.74		
	Graph	0.77	0.77	0.77	0.03	< 0.01
Falling in Love	Best baseline: Content	0.68	0.66	0.63		
	Eugenio et al. [43]	0.67	0.65	0.62		
	Li et al. [61]	0.65	0.65	0.59		
	Graph	0.65	0.65	0.65	0.02	< 0.01
Starting School	Best Baseline Content	0.69	0.7	0.69		
	Eugenio et al. [43]	0.68	0.69	0.68		
	Li et al. [61]	0.68	0.69	0.68		
	Graph	0.71	0.71	0.71	0.02	0.01

We now see that across all our events, our graph approach suggests a compatibility with hypothesis 2 with an increase of F-measure between 0.02-0.04 (median 0.03). In nearly all cases (except Starting School), we see our graph approach scores boosted by feature reduction, vs our baselines, where their scores drop.

5.6.3 Graph Feature Analysis

In this section we discuss the performance of our graph features, showing some of our top performing sub-graphs for our classifiers, as well as discuss how our graph features perform individually. For textual descriptions of our sub-graphs, we use similar notion to the cypher query language[47]. Nodes are contained within (), while directional relationships are expressed as -[]->. E.g. *(my)-[next]->(father)*, would translate to *my father*, with next referring to the relationship of father coming next after my.

5.6.3.1 ConceptNet Analysis

In this section we present the top ten performing features for ConceptNet for each theme.

Table 5.14 shows our top performing concept features for Death of a Parent. As we can see, there are a lot of prominent features with high information gain scores, revolving around synonyms for parental figures such as */c/en/dad*, */c/en/mother*, and */c/en/father*, as well as a more generic */c/en/parent*. We also see terms such as *gone*, */c/en/take_away*, and */c/en/pass* terms associated with with the passing on of someone.

Table 5.15 shows the top performing ConceptNet features for Falling in Love. We see two very strong features such as */c/en/love*, and */c/en/care*, with high information gain scores. We also see concepts such as */c/en/propose_to_woman* appear quite strongly suggesting

Table 5.14: ConceptNet Top 10 Features - Death of a Parent

Feature	Information Gain
/c/en/dad	0.34
/c/en/mother	0.32
/c/en/father	0.27
/c/en/parent	0.26
/c/en/home	0.21
/c/en/gone	0.21
/c/en/far	0.19
/c/en/take_away	0.19
/c/en/pass	0.19
/c/en/distance	0.19

Table 5.15: ConceptNet Top 10 Features - Falling in Love

Feature	Information Gain
/c/en/love	0.30
/c/en/care	0.25
/c/en/give_gift	0.19
/c/en/sing	0.18
/c/en/kiss	0.17
/c/en/marriage	0.17
/c/en/forgive	0.17
/c/en/propose_to_woman	0.17
/c/en/god	0.16
/c/en/kissing	0.16

some posts may have contained proposals in them. Interestingly, we also see concepts such as */c/en/forgive* suggesting some posts could have been apologetic in tone, as opposed to declarations of love.

Table 5.16 shows the top performing concepts for Getting Married. We see a number of terms related to weddings such as */c/en/wedding*, and */c/en/marry*. Possibly a reference to one of our keywords in Section 4.3.1, we see *tying the knot*, */c/en/tie*, */c/en/gordian_knot*, and */c/en/knot*. We also see */c/en/create_from_raw_material*, which is related to the con-

Table 5.16: ConceptNet Top 10 Features - Getting Married

Feature	Information Gain
/c/en/wedding	0.15
/c/en/marry	0.13
/c/en/tie	0.11
/c/en/gordian_knot	0.11
/c/en/knot	0.11
/c/en/half_hitch	0.11
/c/en/create_from_raw_material	0.11
/c/en/hitch	0.10
/c/en/rate	0.10
/c/en/kissing	0.16

Table 5.17: ConceptNet Top 10 Features - Having Children

Feature	Information Gain
/c/en/baby	0.36
/c/en/child	0.33
/c/en/person	0.28
/c/en/young	0.26
/c/en/small	0.26
/c/en/birth	0.23
/c/en/human	0.22
/c/en/born	0.22
/c/en/delivery	0.18
/c/en/infant	0.18

cept knot as well.

Table 5.17 shows our top performing concepts for Having Children. We see strongly performing features that revolve around concepts for babies, such as */c/en/baby*, */c/en/child*, as well as a direct parent to these concepts, */c/en/person*. Interestingly we see the concept */c/en/young*, which could suggest that young is a common relationship stemming from concepts such as */c/en/baby* and */c/en/infant*. We also see some concepts related to the act of giving birth (*/c/en/birth*, */c/en/born*, and */c/en/delivery*).

Table 5.18: ConceptNet Top 10 Features - Starting School

Feature	Information Gain
/c/en/test	0.57
/c/en/gym	0.57
/c/en/computer	0.57
/c/en/school	0.56
/c/en/education	0.56
/c/en/children	0.55
/c/en/class	0.55
/c/en/college	0.55
/c/en/town	0.55
/c/en/building	0.54

Table 5.18 shows our top performing concepts for Starting School. As we can see, all our top features have very similar information gain scores, and revolve around concepts related to schools. Given our performance earlier on in Section 5.6.1 was the weakest for Starting School using Concepts, this might suggest that such strong features might be causing potential over fitting of the dataset.

5.6.3.2 WordNet Analysis

In this section we present our top ten features for WordNet for each of our themes.

Table 5.19 shows our top performing WordNet features for Death of a Parent. Immediately we see a very strong feature, *parent*, with several synonyms related to it (e.g., *mother,female_parent*, and *dad,dada...*). In addition, we also see several synsets related to passing away, such as *away,aside*, and *off, away*.

Table 5.20 shows our top performing WordNet features for Falling in Love. Here, most of our features link quite strongly to the theme. *love*, *lover*, and *romance*, are all strongly related to the event of falling in love. Even *whore* is loosely related to the theme, and most

Table 5.19: WordNet Top 10 Features - Death of a Parent

Feature	Information Gain
parent	0.69
mother,female_parent	0.29
ma,mama,mamma,mom,momma,mommy,mammy,mum,mummy	0.29
dad,dada,daddy,pa,papa,pappa,pop	0.28
father,male_parent,begetter	0.28
away,aside	0.21
off,away	0.21
away	0.21
away,out	0.21
aside,away	0.21

Table 5.20: WordNet Top 10 Features - Falling in Love

Feature	Information Gain
love	0.20
lover	0.15
woman,adult_female	0.13
romance	0.12
person,individual,someone,somebody,mortal,soul	0.11
copulate,mate,pair,couple	0.10
take,have	0.10
like	0.10
whore	0.09
adult,grownup	0.09

likely directly related to terms such as *lover* within WordNet.

Table 5.21 shows our top performing features for Getting Married. Again, most of our features relate heavily to the theme (*bridal,espousal*, *marriage,wedding,marriage_ceremony*, and *ritual,rite*). However most of our other top features appear to relate to the concept of a knot. Besides the obvious ones such as *gordian_knot* and *bow,bowknot*, indirectly related ones include *agglomeration*, and *plant_material,plant_substance*. An agglomeration can be considered a jumbled collection or mass, which is similar to a knot when uses in the context

Table 5.21: WordNet Top 10 Features - Getting Married

Feature	Information Gain
bow,bowknot	0.12
bunch,clump,cluster,clustering	0.12
bridal,espousal	0.12
marriage,wedding,marriage_ceremony	0.12
ritual,rite	0.12
agglomeration	0.12
gordian_knot	0.12
texture	0.12
wood	0.11
plant_material,plant_substance	0.11

of a tight cluster of people or things. As for plant material, we see this is related to knot through a relationship of wood (e.g., wood is a type of plant material, where as a knot is something you get in wood).

Table 5.22 shows our top performing features for Having Children. While we see good information gain scores, all our top features relate in some way to a person. Most of these features relate directly to children (e.g., *child,baby*, *son,boy*, and *daughter,girl*). However, we see our top feature relates to *person*, which could also encapsulate other people entities such as the mother, or father.

Table 5.23 shows our top performing features for Starting School. Similar to our concepts from Section 5.6.3.1, we see a number of very strong synsets that tie directly to concept of a school.

5.6.3.3 Syntactic Analysis

In this section, we take a look at the top ten best performing syntactic graph features for each of our life events.

Table 5.22: WordNet Top 10 Features - Having Children

Feature	Information Gain
person,individual,someone,somebody,mortal,soul	0.47
child,kid	0.45
relative,relation	0.44
offspring,progeny,issue	0.43
organism,being	0.38
causal_agent,cause,causal_agency	0.37
child,baby	0.31
son,boy	0.21
daughter,girl	0.20
male_offspring,man-child	0.19

Table 5.23: WordNet Top 10 Features - Starting School

Feature	Information Gain
educational_institution	0.82
senior_high_school,senior_high,high,highschool,high_school	0.80
learning,acquisition	0.80
secondary_school,lyceum,lycee,gymnasium,middle_school	0.80
institution,establishment	0.79
education	0.79
crammer	0.79
building,edifice	0.77
school	0.76
biological_group	0.75

Table 5.24: Syntactic Top 10 Features - Death of a Parent

Feature	Information Gain
(:)-[has_tag]->(:), (usr)-[next]->(:), (rt)-[next]->(usr)	0.32
prp$	0.23
pass	0.21
(nn)-[nmod:poss]->(prp$)	0.21
my	0.20
awai	0.20
(nn)-[nmod:poss]->(prp$), (nn)-[nmod:poss]->(my)	0.20
(awai)-[has_tag]->(rb)	0.19
vbd	0.19
(vbd)-[dobj]->(nn), (nn)-[nmod:poss]->(prp$)	0.17

Table 5.24 shows our top features for Death of a Parent. Our top pattern can be translated as *RT @username:*, suggesting a large number of tweets were retweets. Considering our negative class is sampled from a Twitter sample stream, this might suggest a large number of those included were retweets. The rest of our patterns all seem to correlate around a similar type of phrase. We see two explicit uni-grams *pass*, and *awai*, in addition to several part of speech tags; *prp$* (possessive pronouns such as my), *nn* (noun such as father), and *vbd* (past tense verb such as passed). Given these definitions, patterns such as *(vbd)-[dobj]->(nn), (nn)-[nmod:poss]->(prp$)* can be pieced together, and would be valid for any of *my father passed*, *my mother passed*, or *my pa passed*. Interestingly, we see this slightly more complex pattern appear above more direct *n*-gram style patterns.

Table 5.25 shows our top 10 syntactic features for Falling in Love. Our most prominent feature appears to be the token *love*, which is similar to our ConceptNet and WordNet analysis. We also see several features associated the previous retweet pattern that we identified for Death of a Parent. We also see several patterns that might relate to statements of love, although generalised with dependency grammar and POS. For example, the last two pat-

Table 5.25: Syntactic Top 10 Features - Falling in Love

Feature	Information Gain
love	0.67
(:)-[has_tag]->(:), (usr)-[next]->(:), (rt)-[next]->(usr)	0.28
(nn)-[amod]->(jj), (love)-[next]->(nn)	0.19
(nn)-[amod]->(jj), (love)-[has_tag]->(nn)	0.19
(:)-[has_tag]->(:), (root)-[root]->(usr), (usr)-[punct]->(:)	0.17
prp	0.16
(nn)-[det]->(dt), (love)-[next]->(nn)	0.16
(nn)-[det]->(dt), (love)-[has_tag]->(nn)	0.16
(nn)-[nmod:poss]->(prp$), (love)-[dobj]->(nn)	0.15
(nn)-[nmod:poss]->(prp$), (love)-[next]->(nn)	0.15

terns contain *nn* (a noun), love, and *prp$* (a personal pronoun, e.g., my), with relationships *nmod:poss* (a noun possessive modifier, e.g., assigning ownership of a noun), and *dobj* (a direct object). Thus a valid example for these patterns could be *I love Jo*.

Table 5.26 shows our top 10 syntactic features for Getting Married. Again, our top pattern relates to *RT @username:*, providing evidence that regardless of theme, retweets are a strong indicator of non events. Besides that, we see the token marri, as a strong feature, as well as several strong syntactic patterns that revolve around phrases like *the knot*. Considering the strong performance of both ConceptNet and WordNet features around *knot*, this makes sense.

Table 5.27 shows our top 10 syntactic features for Having Children. Again, we see our retweet feature top the list. Specifically for having children though, we see patterns such as *(deliv)-[dobj]->(nn)* where dobj refers to a direct object relationship. For example, (deliver, baby), (deliver, child), or (deliver, Steve), all link each noun to the action of delivering.

Table 5.28 shows our top 10 syntactic results for Starting School. Interestingly while we still see the retweet feature as previously mentioned, it is not the best performing feature.

Table 5.26: Syntactic Top 10 Features - Getting Married

Feature	Information Gain
(:)-[has_tag]->(:), (usr)-[next]->(:), (rt)-[next]->(usr)	0.28
marri	0.18
(:)-[has_tag]->(:), (root)-[root]->(usr), (usr)-[punct]->(:)	0.16
(knot)-[has_tag]->(nn), (nn)-[det]->(dt)	0.12
(knot)-[det]->(dt), (knot)-[det]->(the)	0.12
:	0.06
(marri)-[has_tag]->(vbn)	0.06
(nn)-[det]->(the), (nn)-[det]->(dt)	0.06
(:)-[has_tag]->(:), (nn)-[next]->(:)	0.05
(vb)-[next]->(marri)	0.05

Table 5.27: Syntactic Top 10 Features - Having Children

Feature	Information Gain
(:)-[has_tag]->(:), (usr)-[next]->(:), (rt)-[next]->(usr)	0.28
(:)-[has_tag]->(:), (root)-[root]->(usr), (usr)-[punct]->(:)	0.17
(babi)-[has_tag]->(nn)	0.15
(birth)-[has_tag]->(nn)	0.15
nn	0.13
(deliv)-[dobj]->(nn)	0.10
(deliv)-[next]->(nn), (nn)-[det]->(dt)	0.10
(deliv)-[dobj]->(nn), (nn)-[det]->(dt)	0.10
(nn)-[det]->(dt), (babi)-[has_tag]->(nn)	0.10
(nn)-[next]->(nn), (babi)-[has_tag]->(nn)	0.09

Table 5.28: Syntactic Top 10 Features - Starting School

Feature	Information Gain
(school)-[has_tag]->(nn)	0.76
(school)-[has_tag]->(nn), (start)-[nsubj]->(school)	0.37
(start)-[dobj]->(school), (school)-[has_tag]->(nn)	0.37
start	0.36
(:)-[has_tag]->(:), (root)-[root]->(usr), (usr)-[punct]->(:)	0.22
(nn)-[next]->(nn), (school)-[has_tag]->(nn)	0.21
(school)-[has_tag]->(nn), (nn)-[det]->(dt)	0.20
(school)-[has_tag]->(nn), (nn)-[case]->(in)	0.20
(school)-[has_tag]->(nn), (nn)-[amod]->(jj)	0.16
(nn)-[next]->(nn), (school)-[case]->(in), (nn)-[case]->(in)	0.15

Instead, we quite clearly see school, which ties in with our previous analysis of ConceptNet and WordNet, and the strong performance of concepts and synsets around schools. We also see direct patterns such as *(start)-[nsubj]->(school)* which shows the pattern of school being nominal subject of start as a strong feature. Extensions of this pattern such as *(start)-[dobj]->(school), (nn)-[next]->(nn), (school)-[has_tag]->(nn)*, translate to school being the direct object of the verb start, with school either following, or followed by a noun.

Considering our syntactic patterns, we see a difference to our semantics. Our syntactic patterns identify not just the appearance of a token, but the context with which it is used.

5.7 Discussion

In this chapter we have demonstrated a novel way of mining syntactic and semantic graphs around tweets for detecting personal life events. Our results have shown that when compared to the current state of the art, we see a modest boost in F1 of 0.01 across most datasets when considering all features, but a far more significant boost of 0.03 when using feature selection.

Considering our graph features had a boost in F1, after feature selection, merits further work looking at reducing redundancies when mining sub-graphs. Also, given that we mine each feature graph individually, there may exist a number of redundancies that create noise within our feature space.

One surprising feature that we saw in our dataset that appeared strongly was referrals, and the token *rt* which is commonly used to denote a retweet. While our random dataset is probably quite representative of Twitter's usual activity, it does not necessarily mean it is representative of a standard user's timeline. For instance, we have already identified that a number of our tweets seem to have come from commercial users, but the type of user that would benefit most from our work are standard users. Ideally, we need to source our random dataset from existing personal user time-lines, and also ensure that they are annotated, just in case there are too many false negatives within that dataset.

We also had to perform a number of optimisation steps to ensure that we could mine graph patterns efficiently without causing memory issues. This meant that we had to only consider single node patterns across our large semantic networks. Ideally we want to consider multiple node patterns within each of our networks, and see if these types of patterns offer a greater performance boost.

One problem with one of our chosen baselines, Li et al. [61], was that we had to omit the use of conversations because we had only originally collected metrics associated with them in Section 4.3.3. Collecting information like that from Twitter is a particularly tricky task. In addition, we may find that treating the original post, with its conversation, is beneficial to our graph feature set, as well as uni-grams, and content features.

To represent our frequent patterns in our classifier, we serialise our patterns as binary 0/1

to indicate if our pattern exists. However, for larger patterns this might not be as beneficial. For instance, we may find a sub-graph with 5 nodes, and 3 edges, but when counting it against a post, we may find that all the nodes, and two edges match, but the final edge does not. Possibly including sub-graphs with some form of similarity function might increase the usefulness of larger patterns.

When looking at WordNet, we also only considered a standard depth going into WordNet, which included both hypernyms and hyponyms as one. One issue with this is you may find the more specific you get (hyponyms), the more noise you add to your dataset, as its a one to many relationship, vs hypernyms which is a one-to-one. Thus, work needs to be done to look at the effects on performance as we expand each of these metrics. Additionally, there exist other relationships in WordNet such as meronyms, and similarTo. Expanding with these relationships might help enhance our semantic approach.

5.8 Limitations and Recommendations

In this section, we highlight any limitations with the methodology within this chapter, as well as recommendations from this work and suggestions for future work.

5.8.1 Limitations

Whilst we have reported good results with the methodology introduced in this chapter, these need to be viewed in the context of several limitations.

Firstly, we are still using the dataset from Section 4.3.1 which was collected via the use of words contained in WordNet synsets. While we attempted to address some bias highlighted in the previous chapter (4.9.1) with the addition of a randomly sampled negative set, as our

dataset was collected via WordNet synsets, this might have artificially increased the performance of our WordNet features. This is compounded by the additional factor of having small datasets, meaning our classifier might not be representative of all tweets about a particular life event.

Secondly, within this chapter we opted to use only one classifier algorithm, LibLINEAR. Considering the number of features we were looking to compare, this decision was made to reduce the number of permutations of results and experiment runs. The downside to this however is different types of classifiers might have performed better on different feature sets, especially if there was a non linear separation between features and classes.

Finally, while we showed that our frequent sub-graph mining approach worked, there were several limitations due to the size of some of them. Firstly, we had to limit the number of edges per sub-graph to 1. By limiting the number of edges for our sub-graphs, this might have limited the number of interesting patterns being returned. Interestingly, we may have also increased the number of features in our classifiers by implementing this strategy. This is because we were using CloseGraph, which would have filtered out smaller sub-graph fragments with the same support as a larger more connected pattern. Secondly, we mined our different feature graphs independently of each other. An alternative to this approach might have been to join our post tokens to their semantic nodes in ConceptNet or WordNet. An advantage of this would be a possible reduction in redundant features as we would have been mining from a single set of graphs, rather than three different sets of graphs. For example, the token *Wedding* might also have a WordNet synset, and a ConceptNet entry that all only ever appear together. That would mean rather than add the same feature three times in our classifier, we would only represent it once.

5.8.2 Recommendations

With the limitations in mind from the previous section, the work in this chapter has highlighted a novel way of how we can not just detect life events from Twitter, but also discern between event, theme, and un-related tweets across five life events: Death of a Parent, Getting Married, Starting School, Falling in Love, and Having Children. In addition, the frequent graph mining approach should also be scalable for other similar social media classifiers, not just personal life events (e.g., as already shown in Saif et al. [98]).

5.9 Summary

In this chapter, we have demonstrated a novel way of detecting life events from Twitter by expanding individual posts into semantic and syntactic graphs, and then mine frequent subgraphs from them to be used as features in our classifiers. We found that our approaches can perform better than the current state of the art, and can be further improved when combined.

6

Detecting Personal Life Events on Instagram

In previous chapters, our focus has been on extracting personal life events from Twitter. However, as suggested in section 1.1, Twitter is perhaps not the most popular platform for disseminating that form of information[105]. In this chapter, we look at extracting a new dataset from Instagram, adding enhancements to our graph based approach for personal event detection, and re-visit some of our interaction features demonstrated in section 4.4.3.

6.1 Introduction

In our previous two chapters, we have concentrated on generating classifiers for Twitter that identify personal life events, relating to research questions 1, and 2 in section 1.3. However, as suggested in section 1.1, Twitter might not potentially be the best suited social network to find these types of life events on. We also highlighted several issues with how we extracted our original dataset, introducing keyword bias that might be enhancing some of our textual based feature sets. We also could not implement part of Li et al[61] baseline as we did not have conversation text available as part of our original dataset.

We also found several limitations with our graph approach in the previous chapter. We had to perform several optimisation steps to extract frequent patterns, and our approach risked generating a large number of redundant features. In addition, we limited ourselves to single node patterns in our semantic networks, which may limit performance.

This chapter looks to deal with the issues we have just identified, and to that end looks at our final research question, R.Q.3:

R.Q. 3: *Can the techniques used in R.Q.1 and R.Q.2 be used to classify life events on Instagram?*

In addition to this, we propose three additional sub-questions as part of this:

S.R.Q. 3. 1: *- Could the tokenisation of hashtags help improve the performance of our classifiers?*

S.R.Q. 3. 2: *- Could the inclusion of a post's conversation increase the performance of our classifiers?*

S.R.Q. 3. 3: *- Can we enhance our graph based approach by considering the removal of redundant features?*

We also revisit Hypothesis 1, specifically looking at interaction features discussed in chapter 4.

To answer these questions, we first look at extracting several new datasets from Instagram. Due to Instagram's popular use of hashtags, we discover several popular hashtags that are used in our events, and use these to collect our dataset to annotate. As most posts contain multiple hashtags as well as caption text, we can mask our collection hashtags without significantly hindering the performance of our classifiers.

Given our new datasets, we first look at new methodologies that benefit all feature sets we have that can be applied at the pre-processing stage of our pipeline. This includes looking at whether the tokenisation of hashtags, and the inclusion of conversation text, can significantly increase performance. We then focus on optimising our graph features, considering some of the points we mentioned in section 5.7, and consider different varying depths for hypernyms and hyponyms in WordNet, the number of edges we mine from our semantic graphs, and the clustering of our frequent graphs to attempt to reduce noise in our feature sets. Finally, we revisit our interaction features from section 4.4.3, modifying them for Instagram content. Our findings show significant performance boosts across all feature sets for hashtag tokenisations, while modest, but significant boosts when including the whole conversation. We find significant increases (0.01-0.04 for F1) over the state of the art for four out of our 5 events, when considering just our graph approach, and significant increases (0.01 - 0.04 for F1) across all our events when combining our feature sets together.

To summarise:

- We extract a new dataset from Instagram covering the following events: Getting Married, Starting School for the First Time, Graduating, Buying a House, and Giving Birth
- Show the tokenisation of hashtags significantly improves our results
- The inclusion of conversations moderately boosts our results
- Enhance our graph based approach by considering the removal of redundancies in our frequent graph patterns
- Re-visit our interaction features from chapter 4, tailoring them for Instagram content
- Demonstrate a significant increase in performance over our baselines when using our graph approach

The rest of this chapter is outlined as follows. In section 6.2 we introduce the methodology used to extract our Instagram dataset, and enhanced the annotation process. In section 6.3 we go into detail about enhancements to our pipeline process, addressing S.R.Q.1, S.R.Q.2. We address S.R.Q.3 in section 6.4, while introducing our interaction features for Instagram in section 6.5. In section 6.6 we detail our evaluation setup for each of our experiments, and present our results in section 6.7. In section 6.8 we discuss our results, then finally present a summary of this chapter in section 6.10.

6.2 Instagram Dataset

Instagram is a social media site that was developed to share personal photos with both friends and public followers. While the site is focused on the sharing of photos, most photos contain a caption which typically follows the following format:

Caption Text #hashtag #hashtag #hashtag

Our reasons for choosing this site are as follows. First, we believe it to contain more examples of actual events occurring, rather than just thematically being about them. Second, we expect it to be much easier for annotators to work out whether a post is about an important life event or not.

One issue we experienced when trying to annotate Twitter posts was lack of context. With Instagram though, we have additional context to the caption, i.e., the photograph. We can also devise our annotation task to allow annotators to navigate to the post on Instagram to view the full conversation in detail. We believe this additional data should give us a more robust dataset.

6.2.1 Events

In our previous chapters, we focussed on extracting content for *Having Children*, *Getting Married*, *Starting School*, *Falling in Love*, and *Death of a Parent*. However, we found some of these events were quite hard to annotate. For example, annotators struggled with Falling in Love, as it is very subjective and almost impossible to be certain about from a single tweet.

Instead, we have removed two of our previous events, Death of a Parent, and Falling in Love, and selected the next two viable candidates from the list generated by Janssen and Rubin[54], Graduation, and Buying a House.

We also change the names of two of our events to reflect on issues from previous annotations. For *Having Children*, we have changed this to *Giving Birth*. This is to help annotators where a post may simply contain a parents children at any age, rather than when the

actual birth occurred.

Our second event we have decided to rename is *Starting School*, changing it to *Starting School for the First Time*. In our previous annotations, we found a number of posts being annotated as a user starting school when they were entering secondary, graduate, or college, as well as some posts being positively annotated when a child had come back from school holidays. By adding *First Time* to the event title, we aim to make it clearer that we are interested in events where a child may be at school on their first day, or a parent announcing their child has started school.

Thus, our new event list for collection are: *Giving Birth*, *Getting Married*, *Starting School for the First Time*, *Graduation*, and *Buying a House*.

6.2.2 Collection

One of our main issues with our previous dataset was its bias towards the keywords used to collect it, and the inability to remove, or mask those keywords from feature detection. This meant when comparing against a randomly sampled dataset for our negative class, some of our collection keywords would become prominent features.

For collecting our Instagram dataset however, rather than using keywords, we intend to use Hashtags and remove our query ones from our captions. As mentioned in section 6.2, most Instagram posts have an unordered list of hashtags at the end of a post. Thus removing a hashtag from a post will be unlikely to cause issues with its fundamental content.

In order to select useful hashtags, we use Instagrams explore feature [*]. This allows a user to input a hashtag and see if it exists, accompanied with the total number of posts contain-

[*]https://www.instagram.com/explore/

Table 6.1: Instagram Events and Hashtags

Life Event	Hashtags
Buying a House	#homeowners #boughtahouse #firsthome #buyingahouse
Getting Married	#gotmarried #gettingmarried #gettingmarriedsoon #married
Giving Birth	#mybaby #justhadababy #birth
Graduating	#collegegraduate #graduation
Starting School for the First Time	#startingschool #firstdayatschool

ing it. We manually came up with several different hashtags per category, checking their existence and number of total posts using the explore feature. Those which existed on Instagram and had more than 1000 posts are shown in table 6.1.

For each tag, we compared it against Instagram's explore feature[†] to see if it was a viable tag, and roughly how many posts existed per tag. After collecting up to 1000 posts per tag, we randomly sampled 50 of them, and used CrowdFlower to annotate (described in the section 6.2.3). Once these annotations were done, we then selected only those tags that showed a strong possibility of having event posts. Our results for these annotations are shown in figure 6.1.

From the chart, most tags appear to have a large number of events associated with them with the exception of #mybaby. After investigating, we found a large number of these posts appear to use #mybaby to refer to partners, as opposed to a child. Because of this, we dropped #mybaby from further collection.

6.2.3 Annotation

To annotate these posts we use CrowdFlower again, although we modify the questions from our annotation task in section 4.3.2.

[†]https://www.instagram.com/explore/

Figure 6.1: Sample Hashtag Distribution

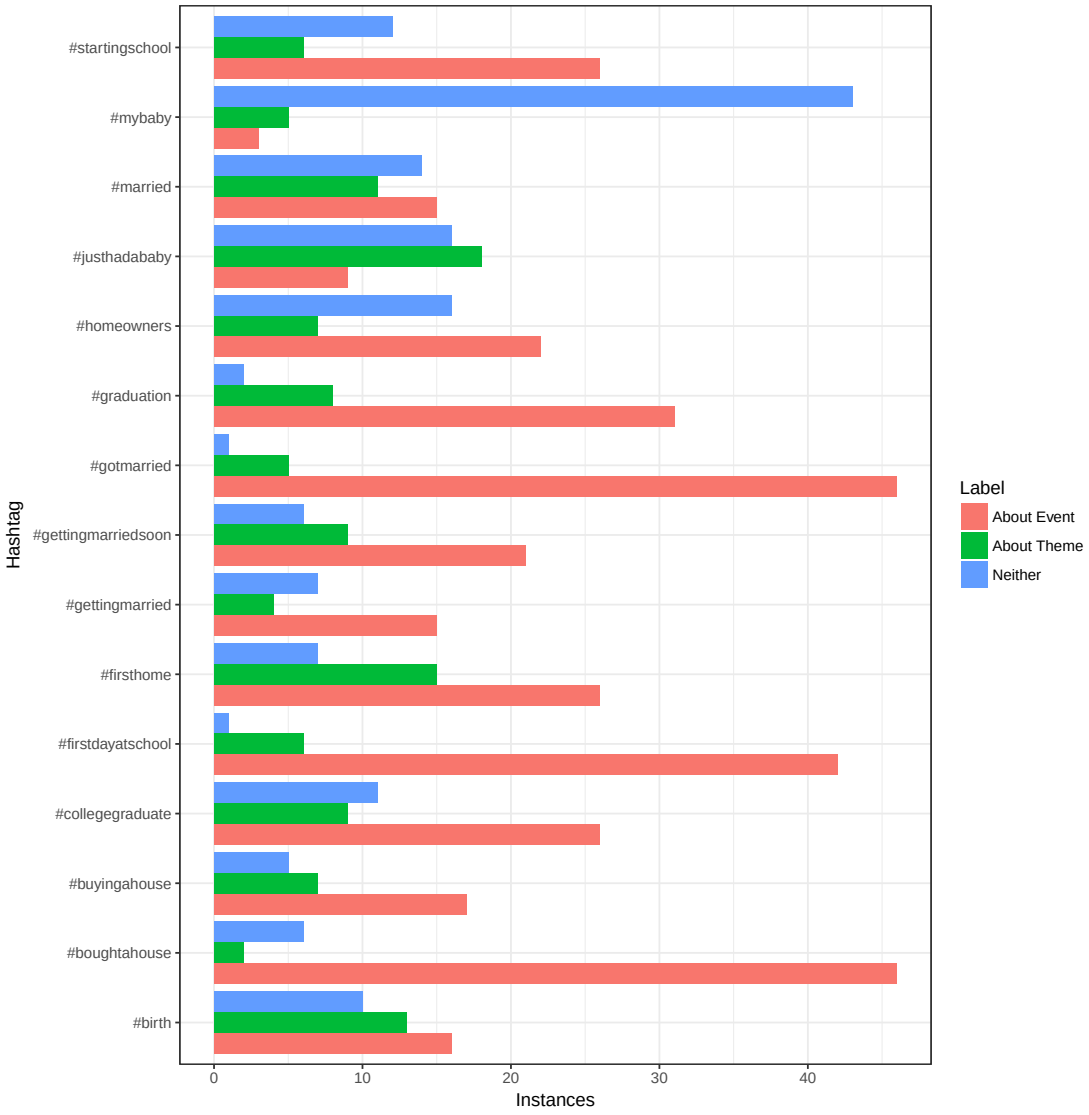


Table 6.2: Instagram Annotation Questions & Answers

Question	Answers
Q1 - Is this post about someone *event theme*?	Yes, someone is Giving Birth No, but the post is related to Giving Birth The post is not related to Giving Birth
Q2 - Is this a personal post, or is it an advert?	This is a personal post This is an advertisement

For each post, we asked users two questions. Table 6.2 shows our questions and possible answers, where *event theme* is replaced with one of our 5 events. Our first question aims to capture whether a post is about an event, thematically related, or has nothing to do with an event. The second is there to deal with filtering Instagram accounts that might be in the same thematic domain, but are companies. For example, we found a number of posts that were to do with selling school lunch boxes, which used tags like #firstdayatschool, to target parents who were sending their children to school for the first time.

Figure 6.2 shows a screenshot of an example annotation question on CrowdFlower.

6.2.3.1 Annotation Results

Figure 6.3 shows the distribution of our final annotations (with adverts removed). As can be seen, unlike our Twitter dataset distribution (figure 4.2), four out of our five themes trend very strongly towards containing posts about events happening, rather than just being thematically related. This might be an indicator that Instagram contains more of these types of events, our chosen hashtags just simply match up well with the types of life events, or that the inclusion of picture provided a more reliable annotation process.

Figure 6.4 shows the distribution of our annotations, but only looking at those posts that annotators thought were either adverts, or related to a company. In contrast, we see the

Figure 6.2: Example CrowdFlower Annotation

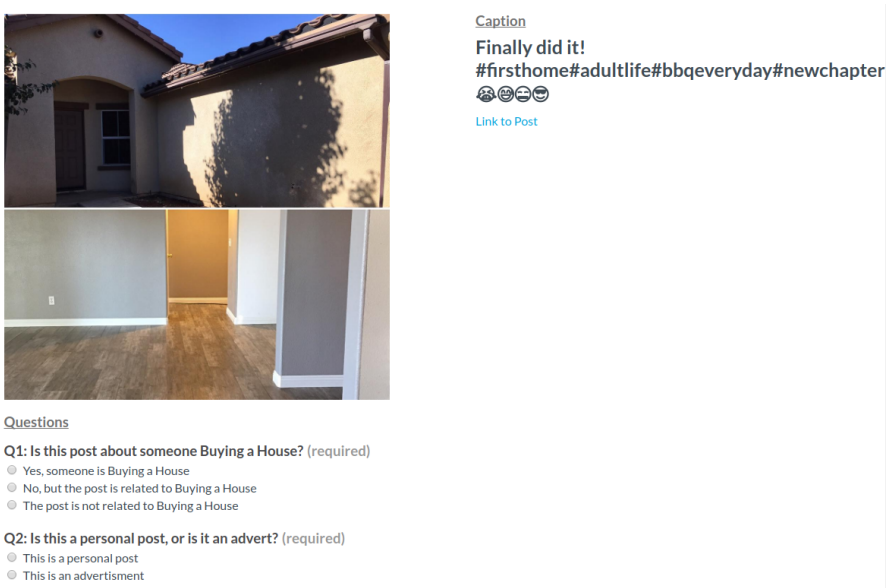


Figure 6.3: Instagram Event Annotation Distribution for Personal Accounts

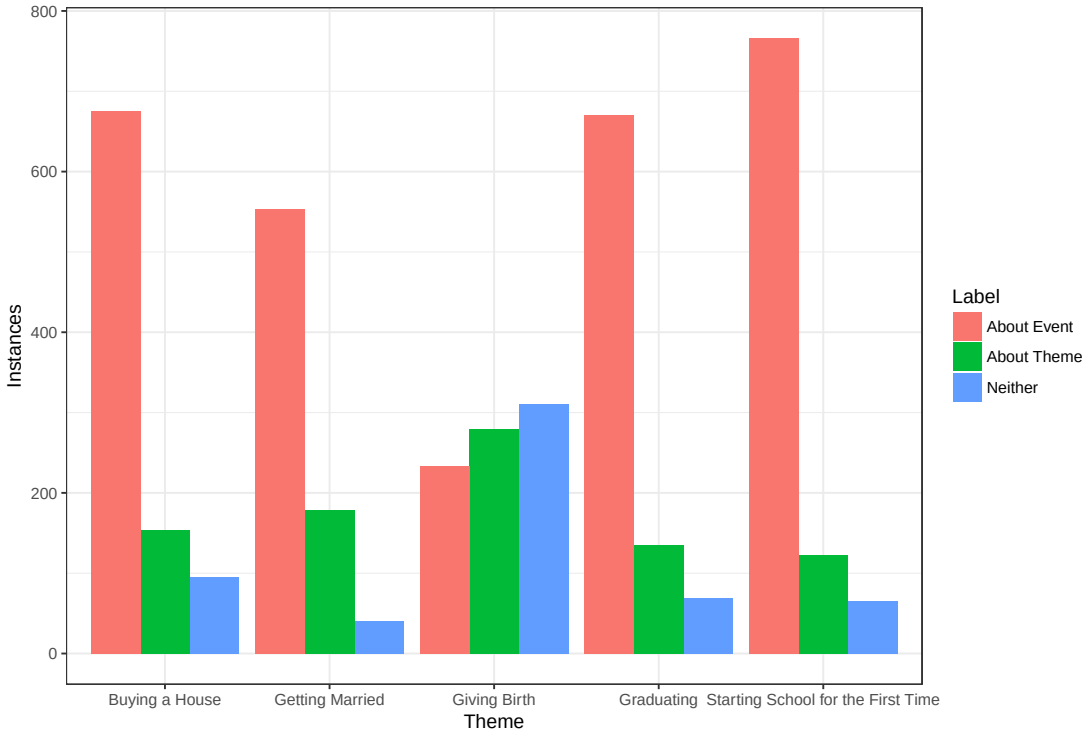


Figure 6.4: Instagram Event Annotation Distribution for Adverts

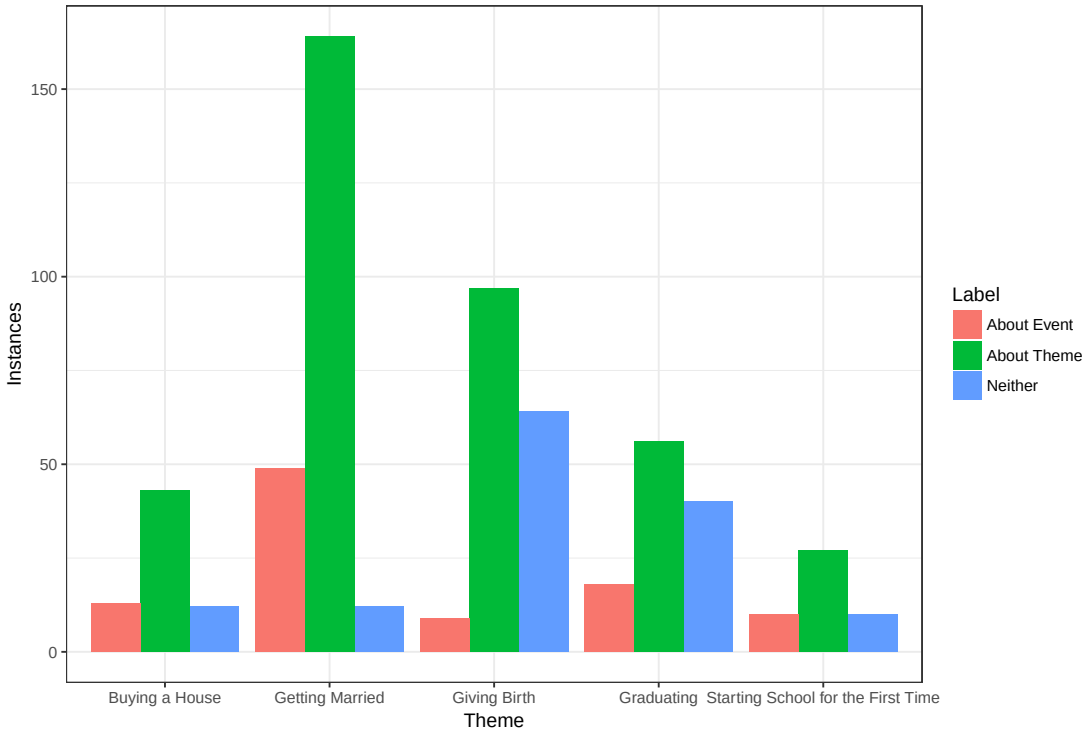


Table 6.3: Instagram CrowdFlower Annotations Agreement

Question	Agreement %
Is this post about event theme?	80.37
Is this a personal post, or is it an advert?	94.17

distribution bias flip in favour of thematic posts.

Table 6.3 shows our agreement metrics across our questions. We see decent agreement scores across both our questions, with a very high agreement score in labelling whether a post is personal, or if it is an advert.

6.2.3.2 Random Instagram Posts

In addition to our set of posts about events, we also collected a series of posts that we could fill our negative class with, without it being too biased to particular events. To do this, for each post that was annotated as about a non advert event, we collected the unique set of users and randomly sampled a separate post from their timeline (collected as part of section 6.2.4).

The task set up was the same as described in section 6.2.3, however we modified Q1 to always be displayed as:

Is this post about someone experiencing an important life event?

6.2.3.3 Random Annotation Results

Figure 6.5 shows the distribution of results, for *personal* posts only, of our annotation task in the previous section. We consider posts that are marked as *neither* to represent posts that are candidates for our negative class in our event classifier. We do not present the results of advert posts, as we only found a total of 40 adverts in our random dataset. The distribution of posts in our *neither* category far outweighs that of our two others.

However, interestingly, we still see about 1/10th of our posts are labelled as being about important life events which could be an indicator that Instagram as a platform, contains more important life events than Twitter.

Alternatively, as our raw dataset comes from users who we already know have posted about previous events, some of these event posts may be related. An event does not necessarily have a one-to-one relationship with a post on an Instagram. Instead, an event may have a series of posts associated with it.

Figure 6.5: Random Annotation Distribution

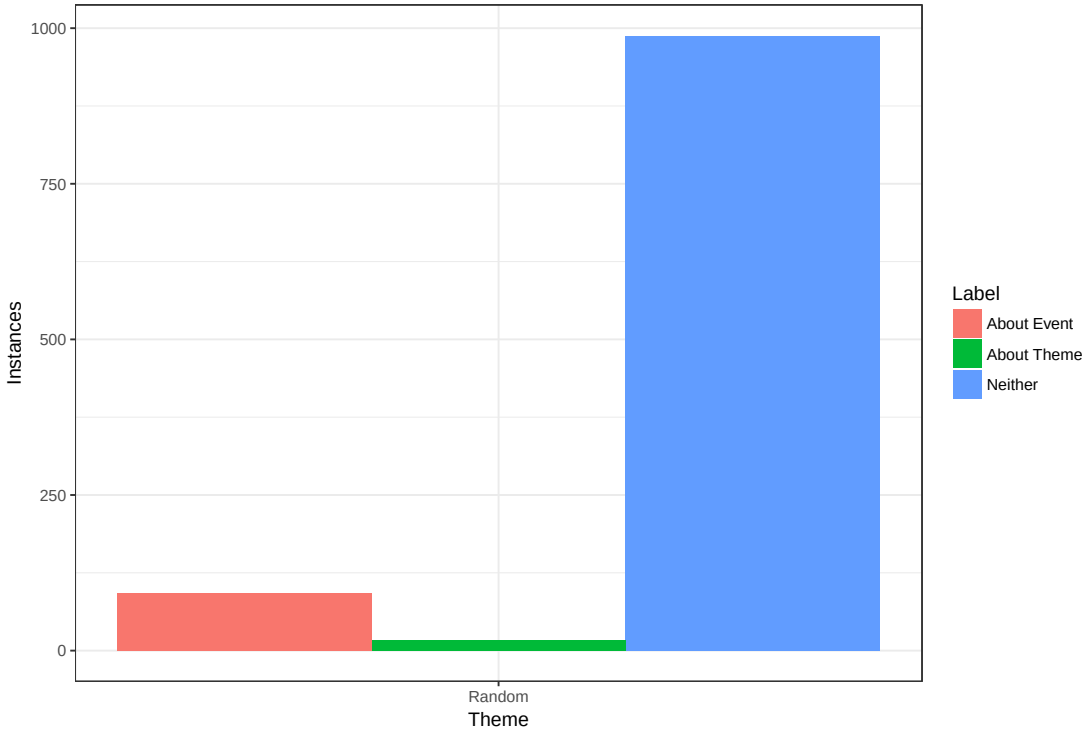


Table 6.4: Instagram CrowdFlower Random Dataset Annotations Agreement

Question	Agreement %
Is this post about event theme?	90.91
Is this a personal post, or is it an advert?	94.7

For instance, a user who gets married may have a post about them arriving at the church, another of them at the alter, and a final one at the reception after the wedding. All of these are about the user experiencing an important life event, thus it would make sense that when randomly sampling our users timelines, we pick up additional posts like these.

Table 6.4 show the agreement rating for this annotation task. As we can see, there is a much higher agreement rating about posts that are about the event theme. This would make sense considering that this should be a much easier annotation task. Also, while the questions are not like for like, in contrast to our best performing trial for our Twitter dataset annotations (table 4.4) we see an improvement over our agreement ratings.

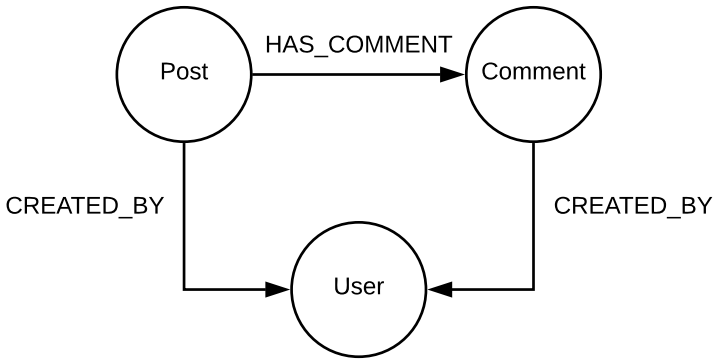
In total, with both the random annotations and the hashtag annotations, our total dataset is composed of 6111 posts.

6.2.4 User Timeline Collection

To enhance our interaction features, we also focus on user timeline collection. For each user in our annotated dataset, we collected all publicly available posts for each user, as well as all conversations for each post. In addition, we collected total number of likes (just the aggregated amount, as Instagram limits retrieving who liked to the most recent 10), and the date the post was created.

This data was extracted by scraping publicly open Instagram profiles using a python web

Figure 6.6: Instagram Graph Design



scraping script [‡]. Data was then stored in a Neo4j graph database[53], consisting of node labels *Post*, *Comment*, *User*, and relationships *HAS_COMMENT*, *CREATED_BY*. Figure 6.6 shows our graph design for this.

6.3 Pipeline Enhancements

In this section we present our enhancements to our pipeline that will affect all feature sets. This includes two enhancements: Hashtag Tokenisation, and Conversations. We explain our hypothesises behind each one, as well as explaining our methodology for including these.

6.3.1 Hashtag Tokenisation

While Instagram is primarily aimed at sharing photographic images, most posts typically have a caption associated with it. To turn each photo into a searchable item within Instagram, hashtags are used to tag the image. However, while a lot of posts have non hashtag

[‡]https://github.com/tomkdickinson/Instagram-Search-API-Python

Table 6.5: Ratio of Tokens to Hashtags

Theme	Hashtags %	Tokens %
Getting Married	0.48	0.52
Buying a House	0.38	0.62
Giving Birth	0.41	0.59
Starting School for the First Time	0.44	0.56
Graduating	0.38	0.62
Random	0.39	0.61
Median	0.4	0.6

text associated with the post, there is a much higher ratio of tags to tokens within each post. Table 6.5 shows this ratio in each of our collected events. We see on average that hashtags take up 40% of Instagram posts.

While hashtags themselves will be captured by considering uni-grams, there are a number of tokens within a hashtag that could potentially be useful.

For example, consider the following common hashtags extracted from our *Getting Married* dataset:

- *#gotmarried*
- *#gettingmarried*
- *#cantwaittobemarried*

Each one of these hashtags are different, however we also note that they are composed of several tokens appended together.

- *#gotmarried* - got married
- *#gettingmarried* - getting married
- *#cantwaittobemarried* - cant wait to be married

In all three of these cases, the token *married* is common, and could thus prove to be a useful feature amongst a number of the feature sets that we are experimenting with.

Thus we present our hashtag tokenisation hypothesis:

Hypothesis 4: *Tokenising hashtags improves the performance of our classifiers.*

6.3.1.1 Hashtag Tokenization Implementation

Hashtag tokenization appears to be an under-researched topic however, to our knowledge, there exists at least one tokeniser with GATE[34], implemented from work done by Maynard & Greenwood[71]. In their work, they develop a hashtag tokeniser that utilises a Viterbi-like algorithm.

Considering that this could be beneficial to a number of our features, we decide to implement this as part of our text processing pipeline from section 4.3.

6.3.2 Conversations

In Li et al.[61], when mining their dataset for topics, they include the whole conversation as part of the document for each tweet. In our twitter dataset, we could not mimic the same behaviour as we had only managed to collect metrics associated with conversations, rather than the conversational text itself.

For our Instagram dataset however, we have managed to extract the conversation for each post in our dataset. Therefore, we intend to represent each post as the text plus the conversational text. We formally present our conversation hypothesis as:

Hypothesis 5: *Including a posts conversation improves the performance of our classifiers.*

6.3.2.1 Implementation

As mentioned, we have collected all conversational data for each post annotated within our dataset. To this end, we intend to concatenate each conversation into our document as a new sentence. Thus, take the following example:

- I love this man more than I can express! So far I really love being married! It's been a great 3 weeks! #gotmarried #ilovemyhusband
    - We got married by him too

As a document, we would then represent this as follows:

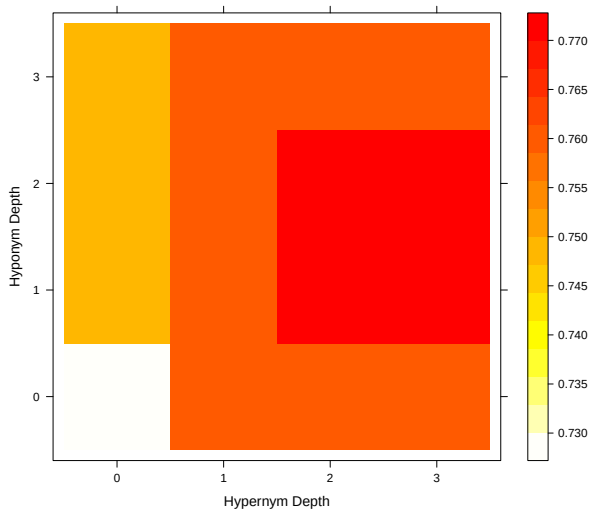
- I love this man more than I can express! So far I really love being married! It's been a great 3 weeks! #gotmarried #ilovemyhusband. We got married by him too

Where a line does not end with a delimiter, we will add a period to indicate the end of a sentence.

6.4 Graph Feature Enhancements

In this section, we present several suggested optimizations to our graph approach introduced in chapter 5. We first consider the different effects of WordNet depths for hypernyms and hyponyms, followed by a clustering method for reducing redundancies within our frequent-graphs to attempt to improve accuracy due to the high volume of patterns we end up extracting.

Figure 6.7: WordNet Depth for Getting Married Against F1 Score



6.4.1 WordNet Depth

Previously in section 5.4.2.1 we looked at the effect of expanding into WordNet, however we only considered depth, regardless of the type of relationship. As stated in our discussion (section 5.7), we may be adding a lot of noise when expanding down in WordNet, vs expanding upwards.

Thus in this section, we wish to explore classifier performance as we expand into hypernyms and hyponyms for one of our datasets.

In order to explore this, we generated 16 sets of features for WordNet, combing different hypernym and hyponym depths between ranges of 0 to 3. For each ARFF file we generated, we then ran Cross Validation 10 times using our LibLINEAR classifier (5.5.4).

Figures 6.7 (Getting Married), 6.8 (Giving Birth), 6.9 (Graduating), 6.10 (Buying a House), 6.11 (Starting School) show our plots for each of our themes. As we can see in each

Figure 6.8: WordNet Depth for Giving Birth Against F1 Score

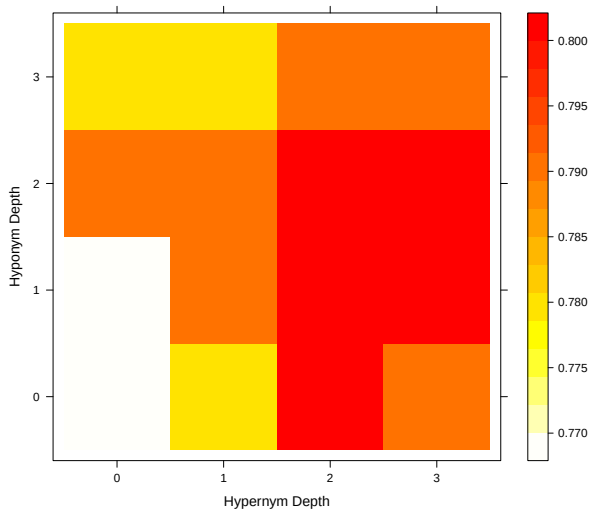


Figure 6.9: WordNet Depth for Graduation Against F1 Score

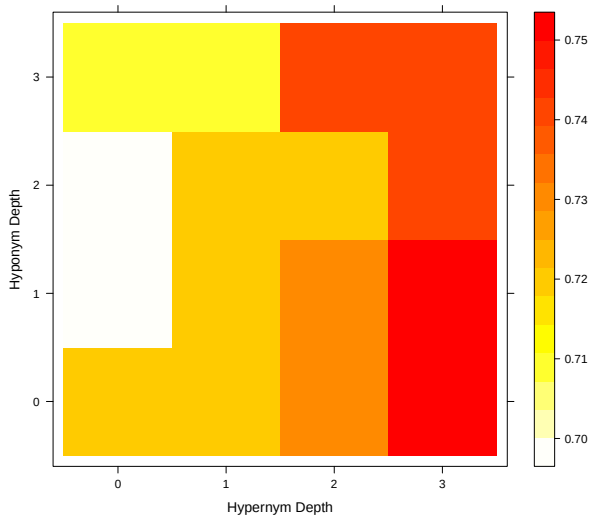


Figure 6.10: WordNet Depth for Buying a House Against F1 Score

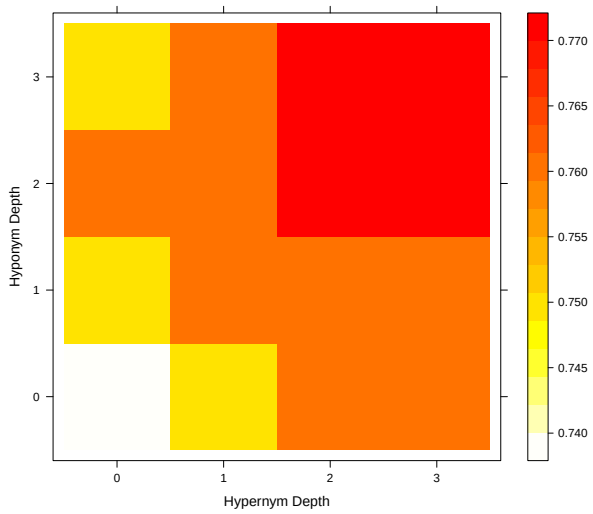


Figure 6.11: WordNet Depth for Starting School Against F1 Score

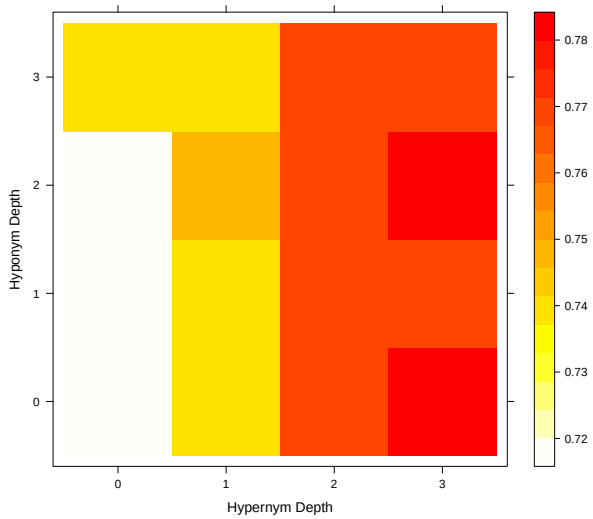
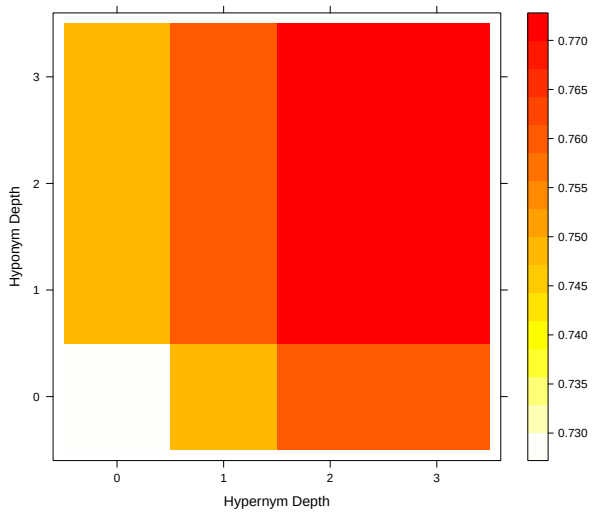


Figure 6.12: WordNet Depth Median Across Events Against F1 Score



figure, expanding just our hyponym depth performs worse than when we expand our hypernym depth. This matches with our hypothesis in our discussion for chapter 5 (section 5.7). As hyponyms are a one-to-many relationship, we end up adding large numbers of patterns to our dataset. Additionally, items are more likely to meet as you become more generic (expanding hypernyms), rather than exact (expanding hyponyms).

Figure 6.12 shows the median effect of hypernym and hyponym depth across all our events. As we can see, a higher hypernym depth is more favourable than hyponym. Considering our other events hot spots, for the rest of our WordNet experiments, we have decided to set our hypernym depth to 3, and hyponym depth to 2.

6.4.2 Clustering of Redundant Patterns

When mining large numbers of frequent graphs, we find that we can include many redundancies within our feature sets. As part of the mining process, we can change our minimum support threshold to a higher value and help reduce the number of mined graphs, however we then end up losing potentially useful sub-graphs that occur at a lower frequency. Alternatively, like in section 5.5.4, we can consider the removal of weak patterns through using a ranking feature selection strategy like information gain. However this can still leave us with many redundancies as many features can be clustered together with similar, if not the same information gain.

To address this within our pattern mining, we consider work done by Xin et al.[125] as a way to cluster our frequent graphs together, and select representative patterns from each cluster.

For this, our pattern distance metric is defined as:

$$Pat\_Dist(P_1, P_2) = 1 - \frac{|T(P_1) \bigcap T(P_2)|}{|T(P_1) \bigcup T(P_2)|}$$

Where P_1 and P_2 are two patterns to compare, and $T(P_n)$ is the set of post ids that particular pattern appears in.

Given this pattern distance, two patterns that co-occur together will have a distance of 0, whereas those who never occur together will have a distance of 1.

We then select a representative pattern per cluster. In our work, we propose to select one pattern per cluster, based on a significance ranking metric, such as information gain, gain ratio, and a baseline of support (i.e., the most frequent pattern).

In order to reduce bias in cross validation, we extract our frequent graphs from our training set, but rank their significance against a separate validation set that is not included in our results.

Thus for this approach, we can use lower minimum support values to identify more frequent sub-graphs, but only use significant patterns that represent a cluster within our mined sub-graphs.

Our hypothesis for this can be represented as:

Hypothesis 6: *Pattern clustering can improve our classification performance on Instagram.*

6.4.2.1 Choice of Delta

When considering clustering our patterns together, we view this as a similar task as selecting our best minimum support value (section 5.4.3). Thus in order to select a threshold for each of our significance measure (support, information gain, and gain ratio), we perform a similar experiment where we consider each feature sets delta threshold against our events.

For each feature set, we mine feature sets with a minimum support of 0.01 (1%), and generate an ARFF file for each choice of delta between 0 to 0.9, in steps of 0.1. We then performed 5-fold cross validation 5 times to obtain our F1 scores, and took the median across each of our five events.

Figure 6.13 shows how delta choice effects our syntactic graphs. We see a steep rise, then a big drop off when we get to around 0.7.

Figure 6.14 shows our delta thresholds for ConceptNet. We initially see poorer results, but then as Delta increase, we see our results improve until about 0.8.

Figure 6.13: Syntactic Graph Delta Thresholds

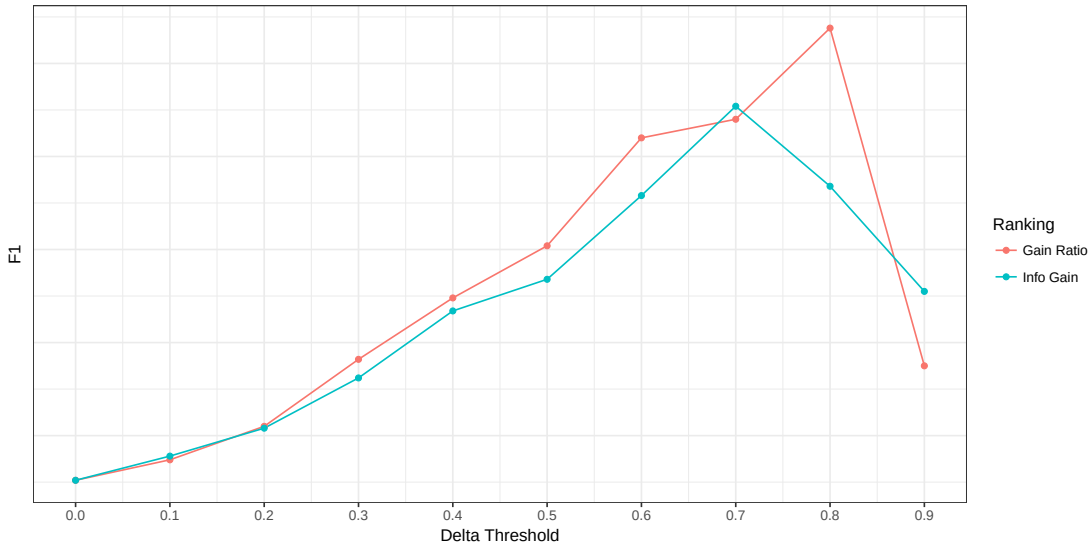


Figure 6.14: ConceptNet Graph Delta Thresholds

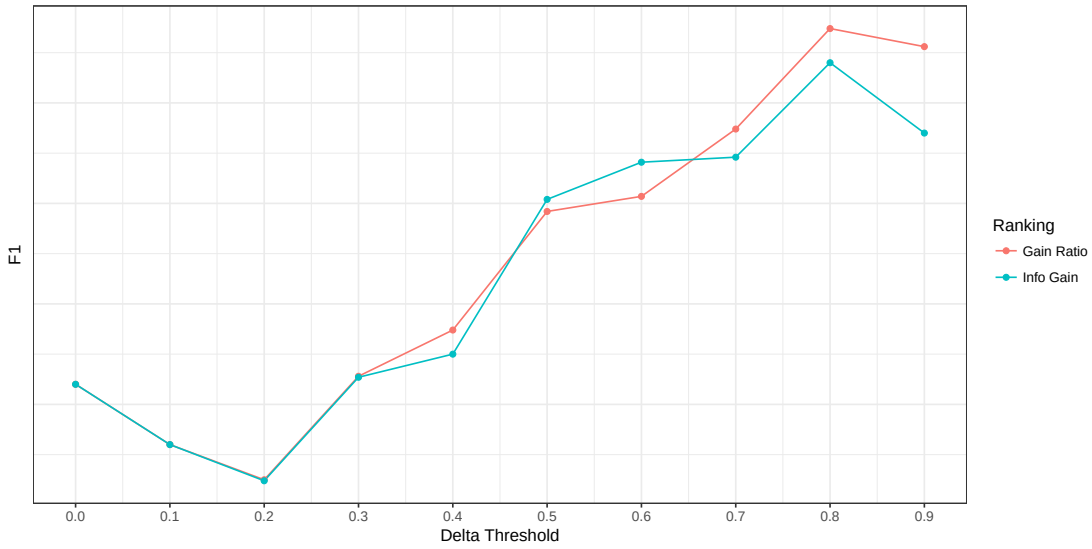


Figure 6.15: WordNet Graph Delta Thresholds

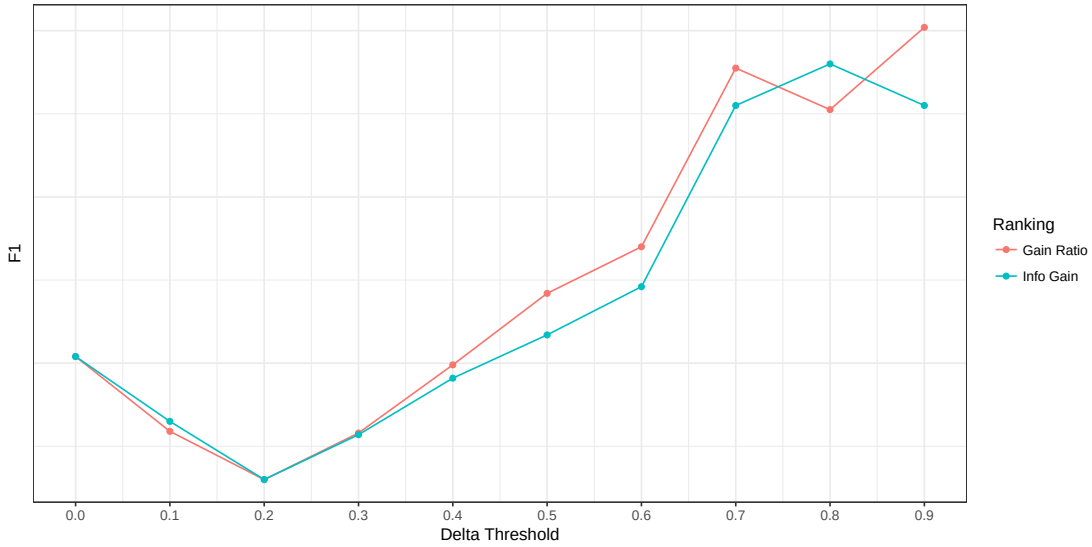


Figure 6.15 shows our delta thresholds for WordNet. Similar to ConceptNet, we see a drop in F1 to begin with, but then it starts to spike as we reduce our feature set further.

For all feature sets, we see good performance at around 0.8, so when we perform delta clustering, we will look at setting delta to 0.8.

6.4.3 Syntactic POS

We have already explored the use of POS within syntactic patterns as part of section 5.4.4. However, the syntactical nature of Instagram posts is expected to be different to that of tweets. Typically tweets, while sometimes abbreviated, represent succinct sentences where grammar plays a role. After viewing our Instagram dataset however, we see a lot of posts dominated by the use of hashtags.

In addition, the POS tagger we have chosen to use is trained specifically on Twitter data. While there are many similarities between tweets and Instagram posts, the syntactic struc-

ture could be considered different between the two mediums. This may incur a penalty on including POS tags on our Twitter data.

Thus, to this end we re-perform the experiment as outlined in section 5.4.4 with our hypothesis as:

Hypothesis 7: *Syntactical POS sub-graphs can improve classification performance on Instagram.*

We report our results in section 6.7.2.1.

6.5 Interaction Features

In chapter 4 we considered interaction features as a viable feature set for our Twitter dataset. We found that while they had some predictability power, they did not significantly contribute to our best performing classifiers. As this is a new dataset from a new social media site, we wish to consider interaction features again.

As mentioned earlier in section 6.2.4, we extracted not just our annotated posts, but also each user's time-line that we have an annotation for.

Our overall hypothesis for interaction features is:

Hypothesis 8: *Interaction features are of use in classifying life events on Instagram.*

Here we present some of our proposed features that use these timelines.

Conversation Time: An Instagram post with a longer conversation time may indicate a post is about a life event, as it triggers a longer conversation versus a standard post. We represent this as the difference in hours from the last comment to the original posts creation date.

Conversation Length: Similar to tracking an Instagram's post conversation time, a tweet about an event may receive a large number of replies in a relatively short time, potentially of a congratulatory, or commissary nature, dependent on the sentiment of the event.

Number of Likes: An Instagram post about a life event may garner a larger number of favourites, or likes, from their followers.

Number of Unique Users: An Instagram post referencing other users might be discussing some form of important event that has happened to them, including those users.

Likelihood of a post gaining *n* likes While the overall *n* of likes might be indicative of a life event, that is an absolute value that has no context to a user's previous number of likes. Here we consider the probability of a user gaining n likes, in contrast to all previous Instagram posts. We calculate this using a Poisson distribution:

$$P(k\ \#\ likes) = e^{-l}\frac{l^k}{k!}$$

Where:

- k - Number of likes on a post
- l - The average number of likes on a user's timeline

Likelihood of a post gaining *n* comments Similar to our previous feature, we measure the relative value and likelihood of a users tweet gaining *n* number of conversa-

tions in the context of all their previous posts. Our hypothesis is that posts about life events will exhibit a spike of number of conversations.

$$P(k\ \#\ retweets) = e^{-r}\frac{r^k}{k!}$$

Where:

- k - Number of retweets on a post
- r - The average number of retweets a user gets across their timeline

6.6 Evaluation Setup

In this section, we present our evaluation setup for each of our experiments as outlined in sections 6.3 (Pipeline Enhancements), 6.4 (Graph Feature Enhancements), and 6.5 (Interaction Features). In addition, we also present our evaluation set-up for comparing against other state of the art baselines.

6.6.1 Dataset

For our evaluation, we intend to build just one classifier for each theme to identify between events and non events. This is mostly due to our analysis of our annotated dataset in section 6.2.3 that shows that most of our posts bias towards about being events. Adding an additional label to try and separate event from theme would likely not perform well for these datasets, as when balancing our classifier, some of our dataset sizes may be under 200 instances in size.

Table 6.6: Classifier Dataset Distribution

Event	+ Event	- Event	Total
Buying a House	675	675	1350
Getting Married	553	553	1106
Giving Birth	234	234	468
Graduating	671	671	1342
Starting School	766	766	1342

Instead, for each event in section 6.2.1 we intend to fill our positive class (+E) with posts that annotators labelled as *Yes, someone is [Event Type].* For our negative class, we intend to fill it with posts from our other two answers for that theme, as well as posts from our random sample, where people answered *No, it is not about an important life event*. In all cases, we also remove any posts that were labelled as being about an advert.

Table 6.6 shows our class distributions for each event.

6.6.2 Classifier

As before in chapter 5 we consider using only one classifier as a control variable to compare each of our feature sets. We use the same classifier as before, LibLINEAR outlined in more detail in section 2.4.4.

6.6.3 Pipeline Enhancements Set-up

In this section we introduce our evaluation set-up for our proposed pipeline enhancements. This includes our hashtag tokenisation (section 6.3.1), as well as the inclusion of conversations in our documents (section 6.3.2). For both experiments, we extract features as detailed in table 6.7 across all events introduced in section 6.2.1.

We then perform 10-Fold Cross validation 10 times, using WEKA's experimenter, and

Table 6.7: Pipeline Enhancement Set-up

Experiment	Feature Set	Description
Hashtag Tokenisation	Baseline (-T)	No hashtag tokenisation occurs
	Tokenisation (+T)	Hashtags are tokenised
Conversation	Baseline (-C)	Only the post text is included
	Conversations (+C)	Conversations are included

consider a result is significantly different using two tailed paired t-tests, reporting significance if $p < 0.05$.

Our results for these experiments are reported later in section 6.7.1.

6.6.4 Graph Feature Enhancement Set-up

In this section we present our evaluation set-up for Clustering Redundant Patterns (section 6.4.2), as well as re-running our experiment for POS in syntactic patterns (section 6.4.3). We have already demonstrated our results for WordNet depth in section 6.4.1, as it was considered a parameter tuning step that was required before doing any other evaluation.

For each of our enhancements, we perform 10-fold cross validation again, over 10 times for each of our graph features, and events. Table 6.8 shows our dataset setup for each experiment.

In regards to delta clustering, we reduce the size of our dataset by introducing a validation set (20% of the training set). This is to independently rank our graph patterns using information gain, and gain ratio, so we do not produce overly influenced results. This allows us to perform 10-fold cross validation without our features being pre-selected against the test data in each fold.

Table 6.8: Graph Feature Enhancement Set-up

Experiment	Feature Set	Description
Tokens Vs POS	Baseline (+P)	We include POS in our syntactic graphs
	No POS (-P)	We remove POS from our syntactic graphs
Pattern Clustering	Baseline (B)	No pattern clustering is performed
	Info Gain (I)	Pattern clustering with using information gain
	Gain Ratio (GR)	Pattern clustering with using gain ratio

6.6.5 Interaction Features Set-up

To evaluate our interaction features, we consider them as a separate feature, and compare them against two baseline classifiers as described in section 3.4.1, ZeroR, and OneR. If interactions perform significantly better than these two features, we will use them along with our graph features to see if they can significantly boost our results.

6.6.6 Baseline

Finally, we look at comparing our best performing methods against the current state of the art. We propose to re-use our baselines from both chapters 4, and 5 (Li et al, Eugenio et al, and Content Features). First, we intend to see if our graph enhancements and interactions, can outperform our chosen baselines. Then, we will consider combining each approach. As before, we combine our three graph feature sets into a single feature, as outlined in section 5.5.2. For both of these experiments, we intend to perform 10 iterations of a 10-Fold cross validation experiment, using Wekas experimenter. We consider a result significantly different if a two tailed paired t-test reports $p < 0.05$. Our results are reported in section 6.7.4.

6.7 Results

6.7.1 Pipeline Enhancement Results

6.7.1.1 Hashtag Tokenisation Results

Table 6.9 shows our results for tokenising hashtags on Instagram. We report F1 scores for both non tokenised (-T F1), and tokenised (+T F1) for each of our feature sets, as well as the change in F1 (Δ F1), and its *P*-value.

As we can see from our results, all features appear to greatly benefit from the tokenisation of hashtags and indicate a compatibility with Hypothesis 4. We see a median increase in of 0.1 across all feature sets.

Table 6.10 provides a summary of the median scores across each of our feature sets. As we can see, both our semantic graphs; ConceptNet, and WordNet, improve significantly with the inclusion of tokenised tags. Topics, and Syntactic both improve by a similar amount, while Uni-grams receive a smaller boost in performance.

Given these results, we intend to use hashtag tokenisation across all our feature sets as way to boost our classifier results.

6.7.1.2 Conversation Results

Table 6.11 shows our results for representing each post as a concatenation of its caption, with its conversation, versus just its caption. We report F1 scores for both without conversation (-C F1) and with conversation (+C F1), as well as Δ F1, and its *P*-value.

As we can see, when including a post along with its conversation, while we mostly appear to see positive results, we do not see a strong compatibility with Hypothesis 5.

Table 6.9: Hashtag Tokenisation Results

Event	Feature Set	-T F1	+T F1	Δ F1	P-Val
Getting Married	ConceptNet	0.61	0.77	0.16	< 0.5
	Syntactic	0.66	0.75	0.09	< 0.5
	Topics	0.67	0.77	0.10	< 0.5
	Unigrams	0.72	0.76	0.04	< 0.5
	Wordnet	0.61	0.77	0.16	< 0.5
Giving Birth	ConceptNet	0.43	0.78	0.35	< 0.5
	Syntactic	0.42	0.72	0.30	< 0.5
	Topics	0.61	0.78	0.17	< 0.5
	Unigrams	0.7	0.80	0.10	< 0.5
	Wordnet	0.64	0.76	0.12	< 0.5
Graduation	ConceptNet	0.57	0.74	0.17	< 0.5
	Syntactic	0.56	0.77	0.21	< 0.5
	Topics	0.67	0.74	0.07	< 0.5
	Unigrams	0.73	0.80	0.07	< 0.5
	Wordnet	0.64	0.75	0.11	< 0.5
Starting School	ConceptNet	0.58	0.75	0.17	< 0.5
	Syntactic	0.71	0.76	0.05	< 0.5
	Topics	0.64	0.70	0.06	< 0.5
	Unigrams	0.70	0.74	0.04	< 0.5
	Wordnet	0.68	0.76	0.08	< 0.5
Buying a House	ConceptNet	0.52	0.78	0.26	< 0.5
	Syntactic	0.71	0.78	0.07	< 0.5
	Topics	0.68	0.78	0.10	< 0.5
	Unigrams	0.77	0.81	0.04	< 0.5
	Wordnet	0.63	0.76	0.13	< 0.5
Average		0.64	0.76	0.10	

Table 6.10: Median Δ F1 for each Hashtag Tokenised Feature Set

Feature Set	Median Δ F1
ConceptNet	0.17
Syntactic	0.09
Topics	0.10
Unigrams	0.04
Wordnet	0.12

Table 6.11: Conversation Results

Event	Feature Set	-C F1	C F1	Δ F1	*P*-value
Getting Married	ConceptNet	0.8	0.82	0.02	< 0.5
	Syntactic	0.8	0.8	0	>=0.05
	Topics	0.79	0.78	-0.01	>= 0.05
	Unigrams	0.81	0.83	0.02	< 0.5
	WordNet	0.78	0.8	0.02	< 0.5
Giving Birth	ConceptNet	0.77	0.78	0.01	< 0.5
	Syntactic	0.76	0.72	-0.04	< 0.5
	Topics	0.78	0.79	0.01	>= 0.05
	Unigrams	0.8	0.8	0	>= 0.05
	WordNet	0.75	0.79	0.04	< 0.5
Graduation	ConceptNet	0.73	0.75	0.02	< 0.5
	Syntactic	0.76	0.76	0	>= 0.05
	Topics	0.75	0.73	-0.02	< 0.5
	Unigrams	0.81	0.81	0	>= 0.05
	WordNet	0.74	0.76	0.02	< 0.5
Starting School	ConceptNet	0.74	0.73	-0.01	>= 0.05
	Syntactic	0.73	0.74	0.01	>= 0.05
	Topics	0.71	0.68	-0.03	< 0.5
	Unigrams	0.73	0.76	0.03	< 0.5
	WordNet	0.73	0.75	0.02	< 0.5
Buying a House	ConceptNet	0.8	0.82	0.02	< 0.5
	Syntactic	0.8	0.8	0	>= 0.05
	Topics	0.76	0.79	0.03	< 0.5
	Unigrams	0.81	0.82	0.01	>= 0.05
	WordNet	0.78	0.79	0.01	>= 0.05
Average		0.77	0.79	0.01	

Table 6.12: Median Δ F1 for each Conversation Feature Set

Feature Set	Median Δ F1
ConceptNet Graph	0.02
Syntactic	0
Topics	-0.01
Unigrams	0.01
WordNet Graph	0.02

Table 6.12 shows our median improvement over each feature set. There appears to be a positive boost for our semantic graphs (0.02 each), but no real boost for syntactic, topics, or uni-grams.

Because of this, for our SOA evaluation, we intend to include conversations only when extracting ConceptNet, and WordNet graphs from our posts.

6.7.2 Graph Feature Enhancement Results

6.7.2.1 POS vs Token Results

Table 6.13 shows our results for including POS syntactic patterns with token features. As we can see, our *P*-values suggest an incompatibility of Hypothesis 7, showing a median reduction of 0.02 in F1 score across our events.

Due to these results, we will be removing POS from our syntactic graphs for our results in section 6.7.4.

6.7.2.2 Pattern Clustering Results

In this section, we present the results for our pattern redundancy experiment. Table 6.15 shows our results, where -C F1 is our F1 without compression, +C F1 is our F1 with compression, Δ F1 the difference between F1s, and its *P*-value.

Table 6.13: Syntactic Graph - Token vs POS Results

Event	No POS F1	POS F1	Δ F1	*P*-value
Getting Married	0.78	0.77	-0.01	>= 0.05
Giving Birth	0.77	0.73	-0.04	< 0.5
Graduating	0.76	0.73	-0.03	< 0.5
Starting School	0.72	0.75	0.03	< 0.5
Buying a House	0.79	0.77	-0.02	< 0.5
Median Scores	0.77	0.75	-0.02	

Our results seem to show that for some themes, compression improves performance, while for others (Giving Birth) it can have a negative effect. However, Giving Birth is our smallest dataset (only 468 instances), so compression at our chosen delta may not be ideal compared to our other themes. We also do not really see much of a compatibility with Hypothesis 6. However, table 6.15 shows the differences in the number of features per feature set. As we can see, while we do not improve F1 score, we provide similar classification results with far fewer results. Whilst we only reduce our feature sets by 25% for syntactic patterns, we see large reductions for WordNet, and ConceptNet (69% and 82% respectively).

Because of the similar performance, but with lower numbers of features, we intend to keep using pattern clustering as we may find when we come to combine feature sets in section 6.7.4, the reduced number of good features, will combine better with other feature sets.

6.7.3 Interaction Feature Results

Table 6.16 shows our results for our Interaction features (section 6.5). As we can see from the table, while performance is low, they appear to be compatible with Hypothesis 8 as they improve on both ZeroR and OneR.

Table 6.14: Pattern Clustering Results

Event	Feature Set	Rank	-C F1	+C F1	Δ F1	*P*-value
Buying a House	ConceptNet	Info Gain	0.79	0.79	0	< 0.5
		Gain Ratio	0.78	0.79	0.01	>= 0.05
	Syntactic	Info Gain	0.8	0.82	0.02	< 0.5
		Gain Ratio	0.8	0.81	0.01	>= 0.05
	WordNet	Info Gain	0.77	0.78	0.01	>= 0.05
		Gain Ratio	0.77	0.78	0.01	< 0.5
Getting Married	ConceptNet	Info Gain	0.77	0.77	0	>= 0.05
		Gain Ratio	0.77	0.77	0	>= 0.05
	Syntactic	Info Gain	0.78	0.77	-0.01	>= 0.05
		Gain Ratio	0.78	0.76	-0.02	< 0.5
	WordNet	Info Gain	0.76	0.76	0	>= 0.05
		Gain Ratio	0.75	0.75	0	>= 0.05
Graduation	ConceptNet	Info Gain	0.75	0.74	-0.01	>= 0.05
		Gain Ratio	0.75	0.74	-0.01	>= 0.05
	Syntactic	Info Gain	0.8	0.8	0	>= 0.05
		Gain Ratio	0.8	0.79	-0.01	>= 0.05
	WordNet	Info Gain	0.74	0.74	0	>= 0.05
		Gain Ratio	0.74	0.74	0	>= 0.05
Giving Birth	ConceptNet	Info Gain	0.77	0.76	-0.01	>= 0.05
		Gain Ratio	0.77	0.76	-0.01	>= 0.05
	Syntactic	Info Gain	0.8	0.79	-0.01	>= 0.05
		Gain Ratio	0.8	0.78	-0.02	< 0.5
	WordNet	Info Gain	0.76	0.75	-0.01	>= 0.05
		Gain Ratio	0.76	0.75	-0.01	>= 0.05
Starting School	ConceptNet	Info Gain	0.74	0.7	-0.04	< 0.5
		Gain Ratio	0.74	0.7	-0.04	< 0.5
	Syntactic	Info Gain	0.74	0.76	0.02	< 0.5
		Gain Ratio	0.74	0.76	0.02	< 0.5
	WordNet	Info Gain	0.75	0.76	0.01	>= 0.05
		Gain Ratio	0.75	0.75	0	>= 0.05

Table 6.15: # Features in Clustered Patterns

Feature	Ratio Type	# Features -C	# Features +C	% Reduction
ConceptNet	gain_ratio	720	224	69
	info_gain	720	226	69
Syntactic	gain_ratio	525	388	26
	info_gain	525	390	25
WordNet	gain_ratio	1737	307	82
	info_gain	1737	302	82
Mean		994	306	59

We see interactions working best for Graduation, and worst for Giving Birth. As these results show Interactions to be a predicative feature set, we will include them in our graph classifier.

6.7.4 Baseline Results

In this section we present our results for comparing our approaches with the state of the art, and present the best classifiers we can produce using our feature sets.

Table 6.17 shows our graph approach, and graph approach + interactions, vs the best performing baseline. For each event type, we report Precision (P), Recall (R), F1, Difference in F1 (Δ F1), *P*-value against baseline (B *P*-value), and the *P*-value of interaction features against our graph approach (I *P*-value).

As we can see from the results, for four of our events, our graph approach, or graph approach + interactions, improves over our results between 0.01, and 0.04. For Buying a House, we see results similar to our baselines, with no significant difference either way.

In most cases, we see a slightly higher precision for our graph method as well (Giving Birth, Getting Married, Buying a House, & Graduating), or the same recall value.

Table 6.18 shows our best performing combination of our syntactic (Syn), semantics

Table 6.16: Interaction Feature Results

Event	Classifier	P	R	F1	*P*-value ZeroR	*P*-value OneR
Buying a House	ZeroR	0.25	0.5	0.33	< 0.05	< 0.05
	OneR	0.51	0.51	0.51	< 0.05	< 0.05
	LibLINEAR	0.57	0.57	0.56	< 0.05	< 0.05
Getting Married	ZeroR	0.25	0.5	0.33	< 0.05	< 0.05
	OneR	0.55	0.55	0.54	< 0.05	< 0.05
	LibLINEAR	0.59	0.58	0.57	< 0.05	< 0.05
Giving Birth	ZeroR	0.24	0.49	0.33	< 0.05	< 0.05
	OneR	0.47	0.48	0.47	< 0.05	< 0.05
	LibLINEAR	0.52	0.51	0.51	< 0.05	< 0.05
Graduation	ZeroR	0.25	0.5	0.33	< 0.05	< 0.05
	OneR	0.57	0.57	0.56	< 0.05	< 0.05
	LibLINEAR	0.63	0.63	0.63	< 0.05	< 0.05
Starting School	ZeroR	0.25	0.5	0.33	< 0.05	< 0.05
	OneR	0.51	0.51	0.51	< 0.05	< 0.05
	LibLINEAR	0.58	0.58	0.58	< 0.05	< 0.05

Table 6.17: SOA Graph Results

Event	Feature	P	R	F1	Δ F1	B *P*-value	I *P*-value
Giving Birth	Best Baseline: Topics	0.78	0.78	0.77			
	Graph	0.79	0.78	0.78	0.01	< 0.05	
	Graph + Interactions	0.79	0.78	0.78	0.01	< 0.05	>= 0.05
Getting Married	Best Baseline: Content	0.8	0.79	0.78			
	Graph	0.82	0.81	0.81	0.03	< 0.05	
	Graph + Interactions	0.82	0.82	0.82	0.04	< 0.05	< 0.05
Buying a House	Best baseline: Topics	0.8	0.8	0.79			
	Graph	0.79	0.78	0.79	0	>= 0.05	
	Graph + Interactions	0.78	0.78	0.78	-0.01	>= 0.05	>= 0.05
Starting School	Best Baseline: Uni-Grams	0.74	0.74	0.74			
	Graph	0.76	0.76	0.76	0.02	< 0.05	
	Graph + Interactions	0.76	0.76	0.75	0.01	< 0.05	>= 0.05
Graduating	Best Baseline: Content	0.79	0.78	0.78			
	Graph	0.79	0.78	0.78	0	>= 0.05	
	Graph + Interactions	0.79	0.79	0.79	0.01	< 0.05	>= 0.05

Table 6.18: SOA Combination Results

Event	Feature	P	R	F1	Δ F1	P-Val
Giving Birth	T	0.75	0.75	0.75		
	I, Syn	0.79	0.78	0.78	0.03	< 0.05
Getting Married	C	0.79	0.78	0.78		
	C, I, Sem, Syn	0.82	0.82	0.82	0.04	< 0.05
Buying a House	C	0.82	0.81	0.81		
	C, Syn, T	0.83	0.82	0.82	0.01	< 0.05
Starting School	C	0.74	0.74	0.74		
	C, I, Sem, Syn, T	0.78	0.78	0.78	0.04	< 0.05
Graduating	C	0.78	0.77	0.77		
	C, I, Sem, Syn, T	0.8	0.79	0.79	0.02	< 0.05

(Sem), interactions (I), Content (C), Uni-grams (U), and Topics (T). As we can see, when combining our feature sets together, we see an improvement ranging from 0.01, to 0.04, over the best performing baselines.

Syntactic features appear to be in every best classifier, while Interactions, appear in four, and semantics appears in three. We then see Content features appear in four of our best classifiers, with topics appearing in three. Our results suggest a compatibility with Hypothesis 3.

6.7.5 Feature Analysis

In this section, we look more closely at our results, and analyse our feature sets for each of our experiments to consider why they performed in that particular way.

6.7.5.1 Hashtag Tokenisation Analysis

In order to understand the uplift in our results after we applied hashtag tokenisation, we looked at the information gain scores of our top twenty features for each feature set. Table

6.19 shows the mean information gain score before we applied hashtag tokenisation, then the information gain score after hashtag tokenisation.

As we can see from the table, in nearly all cases, there is a large positive change in information gain scores after applying hashtag tokenisation. In some cases we see mean information scores improve by more than two times their original value. Only one result, Starting School with ConceptNet, had a negative impact, however it was minor in comparison to the gains ConceptNet features gain in other events.

WordNet also benefits strongly from this increase. For both WordNet and ConceptNet, the increase may be explained that without hashtag tokenisation, no features are extracted from posts that only contain HashTags, or no concepts/synsets in the main text due to the bulk of the message being expressed in HashTags.

6.7.5.2 Conversation Feature Analysis

In section 6.11 we showed that the addition of conversations did not appear to improve our results. In this section we consider why that might be the case.

Table 6.20 shows the difference between average information gain for features with conversations included, and without. We show these values in two scenarios, one with the mean difference in info gain for all features, and one for only the top twenty features. We also show the percent change in dataset size for each event.

As we can see, when considering all features, we see a mostly negative impact on average information gain, but when considering just our top twenty features, a mostly positive change. This is mostly due to the dataset increase, as in all cases we see our attributes increasing in size by an average of 62% (min 41%, max 111%).

Table 6.19: Hashtag Tokenisation Feature Analysis Information Gain

Event	Feature	Baseline		Tokenised Hashtags		% Change
		Info Gain	# A	Info Gain	# A	
Buying a House	ConceptNet	0.0079	397	0.0109	918	38
	Syntactic	0.0015	2324	0.0021	4062	40
	Topics	0.0003	2638	0.0004	2237	33
	Unigrams	0.0014	917	0.0014	1235	0
	WordNet	0.0065	1374	0.0073	2801	12
Getting Married	ConceptNet	0.0052	320	0.0086	938	65
	Syntactic	0.0014	1688	0.0021	3894	50
	Topics	0.0002	2378	0.0004	1923	100
	Unigrams	0.0012	809	0.0014	1120	17
	WordNet	0.0043	1008	0.0052	2573	21
Giving Birth	ConceptNet	0.0071	564	0.0103	1176	45
	Syntactic	0.0013	3470	0.0016	6242	23
	Topics	0.0005	1186	0.0008	946	60
	Unigrams	0.0029	474	0.0029	671	0
	WordNet	0.0029	1766	0.0049	3029	69
Graduating	ConceptNet	0.0029	461	0.0064	1058	121
	Syntactic	0.0008	3710	0.0011	5691	38
	Topics	0.0002	2699	0.0003	2157	50
	Unigrams	0.0011	1053	0.0012	1371	9
	WordNet	0.0014	1791	0.0034	3050	143
Starting School	ConceptNet	0.0089	406	0.0086	945	-3
	Syntactic	0.0015	2733	0.0025	4824	67
	Topics	0.0002	2709	0.0002	2473	0
	Unigrams	0.0012	1094	0.0014	1354	17
	WordNet	0.0049	1331	0.0064	2513	31

Table 6.20: Conversation Information Gain Analysis

Feature Set	Δ Info Gain	Top 20 Δ Info Gain	# Attribute % Increase
ConceptNet	-0.002	0.011	58
N-grams	0.000	0.004	82
Syntactic	-0.001	0.004	111
Topics	0.000	0.005	36
Unigrams	0.000	0.004	41
WordNet	-0.001	0.008	47
Average	-0.001	0.006	62

Considering the lack of significant increase for some of our feature sets, this might suggest that by including conversations, we are adding a large amount of noise to our dataset that hinders classifier performance.

6.7.5.3 SOA Feature Analysis

In this section we look at our best performing features for each of our new feature sets. For each theme, we have ranked the top ten performing features using information gain as our measure.

Table 6.21 shows our best graph features for buying a house. Our typical top features revolve around the concept of a house, with both ConceptNet, WordNet, and Syntactic features all relating to it.

Table 6.22 shows our best features for Getting Married. Our top three performing features are all concepts, related to various things associated with weddings. We also see more generic synsets such as *social event*, and *social gathering, social affair*.

Table 6.23 shows our best features for Giving Birth. Immediately we can see a non English concept, */c/pt/recem_nascio*, which is Portuguese for *New Born*. We see other similar English terms such as */c/en/infant*, *babi*, *baby*, as well as actions that might be attributed to

Table 6.21: Buying a House Best Graph Features

Feature	Information Gain	Feature Type
housing,lodging,living_accommodations	0.18	WordNet
cabin	0.18	Syntactic
/c/en/dwelling	0.17	ConceptNet
house	0.14	Syntactic
/c/en/home	0.10	ConceptNet
location	0.09	Syntactic
unit,social_unit	0.09	WordNet
home	0.09	Syntactic
hous	0.09	Syntactic
condominium,condo	0.09	WordNet

Table 6.22: Getting Married Best Graph Features

Feature	Information Gain	Feature Type
/c/en/wedding	0.20	ConceptNet
/c/en/bride	0.17	ConceptNet
/c/en/ring	0.16	ConceptNet
wed	0.16	Syntactic
marriage,wedding,marriage_ceremony	0.16	WordNet
practice,pattern	0.15	WordNet
/c/en/marriage	0.14	ConceptNet
social_gathering,social_affair	0.11	WordNet
party	0.11	Syntactic
social_event	0.11	WordNet

Table 6.23: Giving Birth Best Graph Features

Feature	Information Gain	Feature Type
/c/pt/recém_nascido	0.20	ConceptNet
chordate	0.20	Syntactic
babi	0.20	Syntactic
child,baby	0.20	WordNet
/c/en/infant	0.20	ConceptNet
baby	0.18	Syntactic
/c/en/cry	0.18	ConceptNet
offspring,progeny,issue	0.16	WordNet
/c/en/seed	0.15	ConceptNet
woman,adult_female	0.13	WordNet

Table 6.24: Graduating Best Graph Features

Feature	Information Gain	Feature Type
/c/en/degree	0.15	ConceptNet
graduat	0.13	Syntactic
/c/en/graduation	0.10	ConceptNet
/c/en/university	0.08	ConceptNet
marriage,wedding,marriage_ceremony	0.07	WordNet
wed	0.06	Syntactic
/c/en/mit	0.06	ConceptNet
/c/en/student	0.06	ConceptNet
colleg	0.05	Syntactic
hacienda	0.05	Syntactic

babies (e.g., */c/en/cry*). We also see a synset referring to *woman,adult_female*, which possibly relates to posts mentioning the mum as well as the child.

Table 6.24 shows our best features for Graduating. We see concepts directly related to universities such as */c/en/degree*, */c/en/university*, and */c/en/graduation*, all with strong scores. Interestingly we see *marriage,wedding,marriage_ceremony*. While its information gain score is relatively low, this could suggest that marriage is a common co-occurring event with graduation on Instagram.

Table 6.25: Starting School Graph Features

Feature	Information Gain	Feature Type
school	0.13	Syntactic
school,schooling	0.13	WordNet
/c/en/pens	0.11	ConceptNet
institution,establishment	0.09	WordNet
biological_group	0.08	WordNet
/c/en/child	0.07	ConceptNet
/c/en/wedding	0.07	ConceptNet
(first)-[next]->(dai)	0.07	Syntactic
wedding,wedding_party	0.06	WordNet
wed	0.06	Syntactic

Table 6.25 shows our best features for Starting School. Our top features revolve around *school*, which is similar to our previous analysis from section 5.6.3, when we considered Twitter. However, we do not see the same high information gain scores. This could either be due to how we sampled our dataset (e.g., removal of keywords), or a difference due to a change of social media sites. Besides school, we also see a syntactic pattern (first)-[next]->(dai), which could come from people mentioning *First day* in posts, or using a hashtag like #firstday. Similar to graduation, we also see a few references to wedding. It would be a lot harder to consider that weddings are common with starting school, so there is a possibility that our negative class may contain more posts about weddings than others.

6.8 Discussion

In this chapter, we have looked beyond Twitter at Instagram, as a new source of identifying personal life events on social media. We have shown that techniques developed throughout this thesis can be applied to other social networks, and saw an average F1 score of 0.8 across five life events.

After highlighting several issues we found with our annotation process in section 4.8, we attempted to correct some of these with our Instagram annotations. Interestingly, as opposed to collecting too many thematic posts in our Twitter dataset, for Instagram we collected too many event posts that made identifying theme first an irrelevant task. Either this is because our collection process focussed on hashtags that were only related to those events happening, or the number of these types of personal events occur much more frequently on Instagram (suggested in Sheldon & Bryant[105]). Most likely it is a mixture of both, as we did find some hashtags that were discounted that may have contained more thematic posts than events (figure 6.1).

Interestingly, we found that the inclusion of conversations did not have a big impact on our results; if anything they actually reduced performance. One cause for this might be because some of our feature detection techniques are susceptible to high dimensionality, known for reducing the performance of classifiers due to over-fitting. A better way to approach this might be to look at building a classifier that can identify condolence/congratulatory posts (e.g. Li et al[61]), and instead of including conversations in our dataset, identify whether posts like that occur within the conversation of a post.

Considering Instagram is specifically a photo sharing website, one route for future research could look at the inclusion of analysis of the images. For example, convolutional neural networks[59] could be used to identify posts based on images. With the captions included, we may see a higher accuracy.

As part of this chapter, we also looked at reinvestigating interaction features from chapter 4. Similar to our first chapter, there is predictability value in these features, but they did not add much to overall classification performance. Given we have experimented against

two different social media sites with these features, our chosen features may not be well suited versus other textual approaches.

As part of this chapter, we also reconsidered including POS in our syntactic patterns. Unlike in the previous chapter (section 5.4.4), we found that including POS seemed to hamper our performance. One particular reason for this, might be the POS tagger that we used was specifically generated for Twitter, not Instagram. To the best of our knowledge, we do not know of any that tailor specifically to the platform. The syntactical structure of Instagram posts might also have a negative effect on the performance of a Twitter specific tagger. While on Twitter, we see maybe one or two hashtags in the text, on Instagram hashtags appear to be used far more frequently, with a large number of tags per post. These syntactical differences are likely to hinder a POS taggers ability to produce meaningful results. An alternative would be to use a generic POS tagger, however these may not handle concepts such as hashtags. As potential future work, we would suggest looking at whether the choice of POS tagger has a significant effect on the performance of our methodology.

One of our best additions to performance in this chapter was the tokenisation of hashtags. Without tokenisation, we found that our average classifier performance across all feature sets performed below average (0.64 F1 score). However, after tokenisation we saw a dramatic improvement across all themes and features, raising our average classifier performance to 0.76 F1 score. This is most likely due to our previous point about the ratio of hashtags to text content on Instagram.

Finally, we looked at the clustering of some of our frequent patterns as outlined in[125]. While we did not see a big performance increase, for our semantic features we saw a large drop in the number of features. This most likely relates to our concern that expanding into

semantic networks may cause a large number of redundancies to be extracted. Given how this technique removed such a large number of features, it would be interesting to see how semantic features performed at greater depths, after clustering occurs.

6.9 Limitations and Recommendations

In this section, we highlight any limitations with the methodology within this chapter, as well as recommendations from this work and suggestions for future work.

6.9.1 Limitations

While in this chapter we have attempted to address the limitations of our acquired dataset from section 4.3, there still exists some limitations to our approach. As hashtags are a categorisation mechanism[§] used to group similar posts together, we may find that we have only selected a few sub-groups of posts for a life event on Instagram, and not a fully representative collection. For example, for *Graduation* we used #collegegraduate, and #graduation. However, these posts might only bring back a subset of posts related to the actual on the day event when a student graduates. While the *Graduation* event might have a single day associated with it, there are other posts related to the event such as receiving acknowledgement from a university that you have passed, or even social occasions such as a graduation ball that might be relevant. This might explain why we found so few posts related to theme, as our hashtag approach was too specific. While collecting a larger number of hashtags is one way to increase the number of posts, a more nuanced way might be to annotate Instagram posts that are a within a window of a positively annotated life event post.

[§]https://help.instagram.com/351460621611097

In regards to our experimental setup, we have again only decided to use a single classifier, LibLINEAR[44]. Our justification for this was to look at reducing the number of experiments to compare individual feature sets against. However, this type of classifier assumes that there is a linear separation between which class a feature might belong to, which might not fit some of our feature spaces. A better approach might be to select the best type of classifier per feature set, and use a multiple classifier system approach such as used by Cavalin et al.[27].

Finally, while we have shown good classification performance in terms of metrics, there are limitations to how well this would perform in a real world scenario. Earlier in section 6.2.3.3 we showed a distribution of a 1000 posts, randomly selected from users who we found posted about a life event on Instagram. While accepting that this sample is small, and not truly random due to the users already reporting at least one life event on their timeline, it serves to highlight one of our original points that on a users timeline, the distribution of posts will see life events appearing in the minority. As we have only tested our classifier on a balanced dataset, we can not suggest that the performance we have shown via metrics, would translate into a real world application.

6.9.2 Recommendations

With the limitations in mind from the previous section, we can provide several recommendations from the work taken within this chapter.

We have demonstrated that with some adjustment, methodologies we developed in chapters 4 and 5 can be applied to classifying life events on Instagram. The events we have shown that are identifiable are Getting Married, Giving Birth, Starting School, Graduating,

and Buying a House.

The most important adjustment is the use of hashtag tokenisation. As demonstrated in section 6.7.1.1, without hashtag tokenisation performance drops across all events for all feature sets between 0.04 and 0.35 in F1 score.

We have also shown how a pattern clustering approach as identified by Xin et al.[125] can reduce the number of features on average up to 59% without a significant decrease in F1 score.

Again, similar to previous chapters, we have shown that a combination of our feature sets increase F1 between 0.01 and 0.04, but highlight that not one single combination provides the best results again recommending that a life event may required a tailored approach.

6.10 Summary

In this chapter, we have demonstrated our ability to extract personal life events from a new social media site, Instagram. We demonstrated significant improvement for all features while tokenising hashtags, and demonstrated improvements to our graph pipeline. We showed that our methodology improves over our baselines with an increase of between 0.01 and 0.04 in F1 measure.

7

Discussion & Future Work

In this thesis we have sought to extend the research and techniques within the area of identifying personal life events by considering our three research questions as outlined in section 1.3. Each empirical chapter (4, 5, 6) has sought to look at our research questions and we have proposed several new methods that enhance the current area of work. In most cases we have seen significant increases against state of the art.

However, we have also experienced challenges throughout this thesis, and thus this section is dedicated to discussing these, and outline future work that seeks to address them. Our discussion section (7.1) begins with Datasets and Annotations (7.1.1), discussing our dataset collection strategies in section 7.1.1.1, and annotation strategy in section 7.1.1.2. We

then present a discussion around our overall classification strategy in section 7.1.2, followed by a discussion of our graph features in section 7.1.3. Our final section for our discussion is section 7.1.4 and focusses on our interaction features from chapters 4 and 6.

We finish this chapter with a section dedicated to future work (section 7.2), and present several new research questions that we can pursue.

7.1 Discussion

7.1.1 Datasets and Annotations

One of our biggest challenges has been dataset collection. For this thesis, we have collected datasets from two different social networks: Twitter (section 4.3), and Instagram (section 6.2).

In section 7.1.1.1 we first discuss our dataset collection strategies. In section 7.1.1.2 we then discuss our annotation strategy applied across these two datasets, and highlight the common issues we found.

7.1.1.1 Dataset Collection

Our Twitter dataset was originally collected with a number of keywords as outlined in section 4.3.1, and after annotation resulted in a dataset of around 14k tweets. We used an approach similar to Eugenio et al[43], but using a much larger mixture of keywords to attempt to obtain a more representative dataset, including various slang phrases, and multiple tenses for keywords. While this collected a number of related posts, one major disadvantage is keyword bias, and may explain why our content features performed far better than others.

One way to avert this would be to mask the collected keywords, but due to a tweet being limited to a set number of characters, we end up losing a lot of information. Related keywords are then lost, and the performance of our classifiers drop significantly. We would argue though that given our annotated dataset, we would prefer a higher precision against a biased set of keywords, than a much lower precision due to important keywords removed.

That is why for this thesis in chapters 4 and 5 we reported our results including our collection keywords. For chapter 6, we used a different collection method, hashtags, which we will discuss in more detail in the next section, however we are not certain how successful this method could be applied to Twitter. One major useful feature of Instagram, is the ability to get a rough idea of how popular a hashtag is. Assuming that Instagram by nature is a more personal platform than Twitter, this approach has worked well. On Twitter though, while there exists services such as Hashtagify[has], a large proportion of our tweets did not contain hashtags. For our Instagram dataset, when taking the median number of hashtags per post we saw 6, and a mean of 7.8. However, for Twitter we see a median of 0 and mean of 0.26 hashtags per post. It is worth mentioning that our Instagram dataset will be biased to hashtag use as we used them to collect posts, however the lack of hashtags on our Twitter dataset indicates that our hashtag method may not be as useful, or representative when collecting posts.

An alternative fix to this problem, specific for Twitter, might be to take the approach that Li et al[61] used, using congratulatory/commiserating phrases in replies to tweets. However, this approach can generalise too much and can cause issues if we are only interested in a few specific types of life events.

Instead, to collect datasets for specific types of tweets we suggest a better approach might

be to combine the two techniques together. Initially we use a broad set of seed keywords to collect datasets for a particular life event. These tweets are then annotated, and those that are positive are then mined for new keywords (using techniques such as topic models, TF-IDF scores, noun phrase detection, etc). Given the set of highly ranked keywords, we can take the compliment against our original keywords, and generate a new dataset to be annotated. This process could be repeated either a limited number of times, or until we see a decline below a threshold in new related keywords. While this technique may still introduce bias, it likely to be far more generalised than our original approach, allowing us to build more representative classifiers, while not sacrificing performance by removing collection keywords.

7.1.1.2 Annotation Strategy

CrowdFlower has been used for annotating both our Twitter, and Instagram datasets. While we sought to address some of our discussion points from section 4.8 about annotation for Instagram, we feel there is more we can do to improve the process.

For our Instagram annotations, one thing we noticed after they were complete, was the number of posts temporally near that were about events. Considering Instagram is a photo sharing platform, this would make sense, and one improvement we could look at would be to include a time window of posts around each collected post, to diversify our dataset.

A step further that could be applied to both Instagram and Twitter, would be to look at annotating entire timelines and profiles. However, we resisted this for this thesis, as that would be an expensive process using CrowdFlower. Some users had thousands of posts on their timelines, and assuming that only a tiny fraction of those were about events, would

make the annotation task more expensive and time consuming.

An alternative would be to get users to annotate their posts themselves. This was an approach we considered early on in the thesis, however the main problem is incentive for users to do this. One option is to create an application that is fun for users to annotate their own posts with, however ethically doing so is tricky (e.g, [22]). One way around this would be to build an application that allows users to tag meaningful posts across their social media accounts, so that they can *bookmark* life events. Given this data, a future iteration of the application could then look at suggesting relevant bookmarks automatically to help generate enough data to make these models accurate.

7.1.2 Classification Strategy

Across our empirical chapters, we considered several different machine learning algorithms, where in chapter 4 we considered Náive Bayes, J48, and Random Forests, while we made the decision in chapters 5, and 6 to only consider LibLINEAR. Our reasoning behind this was we were more interested in comparing the performance of each feature set, and using our choice of classifier as a control variable. However this approach may reduce our overall performance, as different combinations of features may end up performing better with a certain classification strategy.

However, the question then becomes which classifiers would be best suited? This question is why we chose to settle on one classifier for future chapters, otherwise we would be reporting potentially thousands of permutations for our results. An alternative to this though would be to explore algorithms like AutoWeka[112,57], that can perform automatic model selection, and hyperparameter optimisation for a feature set. This would automat-

ically handle the large number of permutations required to consider which classification algorithm handles each feature set best.

7.1.3 Graph Features

In chapters 5 and 6, we presented our graph approach using frequent sub-graphs, to classify our posts, however there a number of challenges and issues with this.

One of our biggest was in the form of overfitting due to the large number of sub-graphs. Tuning our minimum support went some way to help alleviate the problem, but that is a relatively brutal way as we lose less frequent, yet potentially highly correlating patterns. In chapter 6, we reduced this problem somewhat, by using the methodology outlined in Xin et al.[125]. Future work could look at exploring more feature selection strategies such as Fournier-Viger & Tseng[46], where we look to mine the top *k* non-redundant sub-graphs, or consider other forms of feature selection algorithms.

However, a more promising line of work might be to consider other classification strategies for our graphs. For example, there has been recent work that represents graphs to be used in convolutional neural networks (CNN)[113 83], that could be applied to our feature sets. Alternatively we could consider the use of graph kernel based methods[58].

On the topic of our redundancy clustering, we considered only selecting the top performing feature within each cluster. However this leaves open the possibility that certain features are being missed. An alternative to this could be to bucket our features into a cluster. When we come to construct our feature file, we still cluster and reduce the number of features, however we compare every sub-graph within a cluster (e.g. the bucket), until either one matches or we run out. This would allow us reduce the number of features in a

lossless way.

Another area that could improve on is the representation of our sub-graphs in our classifier. Currently, our approach labels a sub-graph as 1 or 0, depending if it appears in a post graph. However, we lose out on larger sub-graphs that may share a high percentage of nodes and edges with the post graph, yet it is marked as 0. We lose the information that it was a close match with this approach. Thus, we could improve on this by introducing different metrics to represent our sub-graphs in our classifiers.

For our syntactic graphs, we used GATE's Twitter model[36] for the tokenisation and tagging of our posts, and a dependency parser from Stanford[28]. However, the success of our syntactic graphs is closely related to how accurate these tools are, and our chosen dependency parser was not specific for Twitter. An alternative could be to use[56] for dependency parsing, and[81] for tokenisation and POS tagging, however due to architecture reasons it was much easier to integrate[28] as part of our solution. Thus an area of improvement might be to consider how these types of parsers affect our syntactic results, especially dependency parsing. In addition, none of these parsers are optimised for Instagram, suggesting an entirely new range of work for the platform.

For our semantic graphs, we only looked at ConceptNet, and WordNet, yet there are lots of other semantic networks that could be used (DBPedia[13], BableNet[78], DBNary[97], WikiData[118] etc). In addition, we also did not consider word-sense disambiguation[117] as part of our graph expansion strategy. This meant that while our graphs recalled most information about each post, their precision was most likely poor. Adding this as a component of our graph expansion would allow us to generate smaller, yet more accurate representations of our posts, hopefully reducing noise in our extracted sub-graphs.

7.1.4 Interaction Features

One of our earliest feature sets were interaction features, which were explored in both chapter 4, and 6. Our hypothesis was that posts about personal life events may exhibit abnormal interactions. Whilst we found they performed poorly by themselves, three of our best performing interaction features (Conversation Length, $P(\textit{Favourite})$, and $P(\textit{Retweeted})$), regularly appeared in our top performing features when combined. This suggests that by themselves, the classifiers suffered from underfitting, although exhibited some good features. An expansion of interaction features may increase their performance, especially when combined with other feature sets.

For example, while looking at the overall relationship between the number of likes or retweets a post gets, and the average number of likes/retweets a post gets, we could consider a similar comparison but over a smaller window of time. Most likely, we would assume earlier posts in a user timeline will have far fewer interactions due to a reduced number of followers. As the account matures though, their follower count goes up, and newer posts have a greater chance of being liked/retweeted. However, if we considered a window, or snapshot, around a post to be classified, this would give us a more relative comparison of whether there is an abnormal interaction activity against for that post.

7.1.5 Motivation

In chapter 1 we outlined the motivation for pursuing this thesis. In this section, we reflect back on the outcomes of the work, and if anything contributes to what was outlined originally in section 1.1.

One of our original motivations was to help sort through our social media history for

personal reflection, identifying what we might be sharing about ourselves with the world. Given that we have built models for seven life events across Twitter, and Instagram data, we could argue that what we have produced can contribute towards that goal, however this would come with several caveats.

Firstly, we do not know how well the models we have produced would work in a real world scenario. Metrics by themselves only show how well our classifiers work on the snapshot of data that we took at the time. Indeed, some of the datasets that were collected are several years old now, and there is a chance that patterns of usage across these platforms might have subtly changed. As already pointed out in section 7.1.1, our dataset collection strategy might also not have retrieved a fully representative set, which may lead to completely different results in the real world. In addition to this, while we have shown good metrics, all we can say with confidence is that we have performed better than our baselines. Without user experiments, we do not know whether our models are performant enough for real world usage.

Secondly, because we have only targeted several life events from the work done by Janssen & Rubin [54], we do not know if users would actually be interested in the subset of events we have targeted. However, while this thesis might not have produced models to cover every type of life event, the methodology within the work can be repeated on datasets collected for new ones as needed.

In fact, we could suggest that single model classifiers are actually more beneficial to our second motivation which was marketing. A marketing strategy can easily be pursued on knowing just one life event, e.g., advertising baby products when someone has had a child.

Our final motivation we suggested was usage within social opportunities, such as online

cyber-bullying. While it would not be considered a life event, the work in this thesis could be adapted to look at identifying either posts, or possibly user profiles of suspected children who are being bullied. A hypothesis could be that aggressive or bullying comments on posts would share syntactic and semantic patterns, as well as the timeline interactions of bullies. As opposed to identifying individual posts, the usage of such as tool would probably be to identify accounts instead. In terms of this capability, we have already shown that the work within this thesis can be adapted to account classification with the work done in Saif et al. [98].

7.1.6 Twitter versus Instagram

Within this thesis we have considered detecting life events on two separate social networks, Twitter and Instagram. In this section, we discuss the differences between these two platforms in the context of the methodology used in this work.

First, let us consider some of the possible differences that might affect the effectiveness of features on the two platform. Table 7.1 compares several different syntactic metrics between Twitter and Instagram for the shared positive life events identified in our datasets: Getting Married, Having Children, and Starting School.

As we can see, there are clearly some stark differences between the two platforms that might have an affect on how useful some of the feature sets we've used in this thesis are. The most obvious difference is the average number of hashtags used per post. Considering one of our top contributions in chapter 6 was the use of hashtag tokenisation, it is doubtful whether that technique would be effective on Twitter.

Even from an interaction point of view, there will be fundamental differences that might

Table 7.1: Syntactic Differences Between Instagram and Twitter

	Social Media	Twitter	Instagram	Difference
	Median Tokens	17.00	18.00	1.06
	Median Characters	108.00	133.00	1.23
Getting Married	Median Hashtags	0.00	7.00	7.00
	Median Hashtags where exist	1.00	7.00	7.00
	Median Tokens	21.00	25.00	1.19
	Median Characters	120.00	170.00	1.42
Having Children	Median Hashtags	0.00	7.00	7.00
	Median Hashtags where exist	1.00	7.00	7.00
	Median Tokens	15.00	14.00	1.07
	Median Characters	80.00	97.00	1.21
Starting School	Median Hashtags	0.00	4.00	4.00
	Median Hashtags where exist	1.00	4.00	4.00

influence any patterns found within interactions. For example Twitter users follow a user's timeline and you will see posts from these users on your home page. However on Instagram you have the ability to follow both a user and a hashtag. Given a popularly followed hashtag is used in a post, this would then promote said post to a far greater number of people than that on Twitter. As the number of views a post has is likely to influence how many likes or comments that post has, it is not inconceivable that would have an affect on the patterns of interaction across the two platforms.

We can also see differences between the two platforms from the point of view of the content submitted to them. While we omitted Falling in Love due to the subjective nature of the event, Death of a Parent was also not included for Instagram. Our reason for this was partly due to our collection strategy where coming up with hashtags to represent Death of a Parent became a difficult task. While some such as #mydaddied existed, the number of posts found were tiny, and most were joke posts. However, in contrast *Death of a Parent* was our second largest dataset from when we collected Tweets.

Interestingly, after our experiments were completed in chapter 6 we discovered that by targeting remembrance, you can find posts related to the death of a parent. When using the *#missmydad* and *#missmymum*, Instagram reports up to 118k and 23k posts respectively. There is an argument to be made though that even using such a strategy on Instagram, you're effectively targeting a sub-life-event, where while someone's parent has indeed died an application would bias towards returning posts only about the remembrance of such an event.

7.2 Future Work

Given our discussion, in this section we present several potential research questions that highlight our plan for future work on the detection of personal life events.

R.Q. 1: *Could considering temporal features across semantic, syntactic, and interactions, help improve the accuracy of our detection methods?*

As we already alluded to in the previous section, one potential area of future work would be to consider the temporal patterns in the lead up, and aftermath, of personal events. We hypothesise that, across semantic, syntactic, and interactions, there exists related posts to the event, especially in the context of Instagram.

For semantic, and syntactic features across certain events, we expect temporally close posts to be related to the event. For example, with Getting Married we suggest that there exist a number of posts talking about the wedding, or honeymoon. For Graduation, we may see a series of posts that highlight passing final exams, followed by posts about new jobs (suggesting a certain common chain of life events).

We may also find that there exist windows of grouped posts about events. Up until this point, our approach has been to classify a single post about being about a life event. However, especially on sites like Instagram, we may find that there exist a group of related posts, all relating to the same event. The task becomes less about identifying a single post, and more about identifying clusters of posts related to a specific life event.

In the context of interaction features, we found for Twitter, considering the probability of a post getting *n* likes or retweets, in respect to the entire timeline worked quite well. However this approach lacks nuance, as we would expect that earlier posts have very few likes and favourites, where later posts are viewed by more followers. Thus we could look at interactions around a window of a target post, and observe whether there is an abnormal number of interactions in the leadup, and after.

R.Q. 2: *Can the representation of our sub-graphs improve the accuracy of our classifiers?*

Up until now, our representation of sub-graph features have been binary in nature, suggesting a hard yes/no as to whether they appear within a post. Yet this disadvantages those features who may have a high level of similarity, but do not match. Larger sub-graphs would be more prone to this, and thus we suggest that an alternative way of approaching it would be to consider a metric that returns how similar a sub-graph is to the post.

A simple percentage of intersecting nodes and edges would serve as a starting point, although some form of normalisation would have to occur. For example, a sub-graph with two nodes with one in common would match at 50%. However a large sub-graph with 6 nodes and one missing, would match at 83%. Thus we could consider several strategies to counter this.

In addition to this, we could also consider adding weights to our semantic feature graphs, based on how far each node is expanded to. After sub-graph mining, we can consider the statistical measures of those features, factoring them into our sub-graph similarity, with the intention to adjust scores based on how far the sub-graphs nodes were found.

As part of this research question, we could also consider the implementation of our bucket suggestion in section 7.1.3, when clustering redundant patterns together. This would help ensure a lossless feature compression, after compressing clusters of our frequent patterns together.

R.Q. 3: *Would other classification techniques such as graph CNNS, graph-kernels, or automatic hyper parameter and model selection algorithms, improve our results?*

As mentioned in section 7.1.3, our approach has been to consider mining frequent sub-graphs to use in conventional machine learning classifiers. However there are several issues with this. Firstly, we see a large amount of redundancy, potentially causing over-fitting. Secondly, mining frequent sub-graphs is an expensive process, and requires large amounts of processing to perform effectively.

In addition to this, for all our graph features, we have only considered using a LibLINEAR classifier. While this algorithm has been shown to perform very well, we suggest that the methodology we have used for building these classifiers should not be tied to a specific classifier.

Thus, this research question seeks to investigate other forms of classification algorithms. In the case of graph CNNs, and graph-kernels, this would only apply to our generated graph feature sets, syntactic and semantic graphs due to their specific support for graphs.

For all other combinations, we can also compare against the best selected feature combinations using AutoWeka.

8

Conclusion

The main goal of this thesis has been to expand the research on detecting personal life events across social media. The state of the art targeting this particular niche problem has been under researched, only targeting Twitter. To this end, we considered three main research questions as part of this thesis:

R.Q.1: What features would improve the classification of important life events?

R.Q.2: Can the inclusion of semantic and syntactic patterns improve performance in our life event classifiers?

R.Q.3: Can the techniques used in R.Q.1 and R.Q.2 be used to classify life events on Instagram?

We addressed each of these research questions in turn, across chapters 4, 5, and 6. This has resulted in considering interactions, semantics, and content features in chapter 4, semantic and syntactic features in chapter 5, and finally using our approaches against Instagram data in chapter 6.

Our main conclusion has been that we can identify life events with reasonably good accuracy, across different social media sites, and that the use of syntactic and semantic graph features show good promise in producing powerful classifiers.

The rest of this chapter presents a conclusion for each of our research questions.

8.1 Interaction, Content, and Semantic Features

In chapter 4, we sought to address our first research question:

R.Q. 1: *What features would improve the classification of important life events?*

To this end, we considered the use of interaction, content, and semantic features to see if these would improve classification performance against a uni-gram baseline by Eugenio et al[43]. As part of this chapter we collected a new dataset from Twitter for the following personal events: Getting Married, Falling in Love, Having Children, Death of a Parent, and Starting School. To collect these tweets, we manually selected several core synsets from WordNet for each event, and included related synsets as keywords for our query. From these results, we annotated over 14k tweets using CrowdFlower[cro].

With this annotated dataset, we then proceeded to build classifiers for each of our feature sets using Naive Bayes, J48, and RandomForests. Our results showed that as individual classifiers, the approach outlined by Eugenio et al, performed similarly to our content feature approach, while interactions and semantics performed poorly by themselves.

After combining our features together though, we saw an increase of 0.003, and 0.01 in F1 measure, although concede that these improvements are minimal.

Whilst interaction features performed poorly by themselves, we did find that three of the chosen features were some of our best performing in regards to information gain.

8.2 Semantic and Syntactic Graphs

In chapter 5, we sought to address our second research question:

R.Q. 2: *Can the inclusion of semantic and syntactic patterns improve performance in our life event classifiers?*

We used our previous dataset from chapter 4, although enhanced it by extracting 2.5k tweets from Twitters sample endpoint to act as a negative class, similar to Li et al. [61].

Given this dataset, we then looked at creating classifiers for the same five life events from our previous chapter. However, rather than a binary classification indicating if a tweet was about an event or not, we considered a trinary classifier instead. Our classes were: +E+T (About Event and Theme), -E+T (Not about event, thematically related), and -E-T (Not about event or theme).

For our new feature sets, we looked at converting posts into both syntactic and semantic graphs using four approaches: tokenisation, dependency parsing, ConceptNet, and WordNet. Given a set of these graphs, they were mined for frequent sub-graphs, with the output used as a feature set.

Using our graph method by itself, saw a significant improvement in three out of five our themes (between 0.02, and 0.04 in Δ F1 measure) over three baselines: content features, Eugenio et al [43], and Li et al [61].

8.3 Identifying Events on Instagram

In chapter 6, we sought to address our final research question:

R.Q. 3: *Can the techniques used in R.Q.1 and R.Q.2 be used to classify life events on Instagram?*

To this end, we first extracted a new dataset from Instagram, considering some of the lessons learned from chapter 4. For this chapter, we decided on the following events: Having Children, Starting School, Getting Married, Graduating, and Buying a House. To collect our new dataset, we implemented a hashtag collection strategy that would allow us to remove the collection tags when processing each post. We experimented with several different hashtags for each event, extracting a total of 6k posts which were all annotated using CrowdFlower.

Given this dataset, we then looked at enhancing our semantic and syntactic graph approach from chapter 5. First, we looked at tokenisation of hashtags which saw a significantly large improvement across all feature sets (between 0.04 and 0.17 in Δ F1 measure, with median average of 0.1 Δ F1). We then considered the inclusion of conversations as part of our feature sets, an approach used in both Li et al. [61], and Cavalin et al. [27], although found that conversations did not offer a large impact on improving results (median of 0.01 in Δ F1 measure).

As part of chapter 5 we were concerned over-fitting may have been an issue due to the large number of features extracted. We addressed this as part of chapter 6 by considering compressing our frequent patterns into clusters, and while we didn't see a significant change in results, we did see a significant drop in the number of features while retaining

good results (mean 59% reduction). We also revisited interaction features from chapter 4.

When using our graph approach with interactions, we saw a significant improvement across four out of five events, and saw a significant improvement across all events (an improvement between 0.01 and 0.04 Δ F1) when combining our feature sets.

References

[cro] CrowdFlower. http://crowdflower.com/.

[tex] TextRazor. https://www.textrazor.com.

[ino] An apriori-based algorithm for mining frequent substructures from graph data.

[fac] Facebook. https://www.facebook.com/.

[has] Hashtagify. https://hashtagify.me.

[ins] Instagram. https://www.instagram.com/.

[twi] Twitter. https://www.twitter.com/.

[ver] Verbix. http://www.verbix.com.

[wik] Wikipedia. https://www.wikipedia.com/.

[10] Agarwal, R., Srikant, R., et al. (1994). Fast algorithms for mining association rules. In *Proc. of the 20th VLDB Conference* (pp. 487–499).

[11] Allan, J. (2002). Introduction to topic detection and tracking. In *Topic detection and tracking* (pp. 1–16). Springer.

[12] Atefeh, F. & Khreich, W. (2015). A Survey of Techniques for Event Detection in Twitter. *Computational Intelligence*, 31(1), 132–164.

[13] Auer, S. o. r., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., & Ives, Z. (2007). DBpedia: A Nucleus for a Web of Open Data. In *Proceedings of the 6th International The Semantic Web and 2Nd Asian Conference on Asian Semantic Web Conference*, ISWC'07/ASWC'07 (pp. 722–735). Berlin, Heidelberg: Springer-Verlag.

[14] Aung, W. T., Myanmar, Y., & Hla, K. H. M. S. (2009). Random forest classifier for multi-category classification of web pages. In *2009 IEEE Asia-Pacific Services Computing Conference (APSCC)* (pp. 372–376).: IEEE.

[15] Bandyopadhyay, D., Huan, J., Liu, J., Prins, J., Snoeyink, J., Wang, W., & Tropsha, A. (2006). Structure-based function inference using protein family-specific fingerprints. *Protein Science*, 15(6), 1537–1543.

[16] Becker, H., Naaman, M., & Gravano, L. (2011). Beyond trending topics: Real-world event identification on Twitter . *Icwsm*, (pp. 1–17).

[17] Berntsen, D. & Rubin, D. C. (2004). Cultural life scripts structure recall from autobiographical memory . *Memory & Cognition*, 32(3), 427–442.

[18] Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition.

[19] Bojanic, D. C. (2011). The impact of age and family life experiences on Mexican visitor shopping expenditures. *Tourism Management*, 32(2), 406 – 414.

[20] Bowyer, K. W., Chawla, N. V., Hall, L. O., & Kegelmeyer, W. P. (2011). SMOTE: synthetic minority over-sampling technique. *CoRR*, abs/1106.1813.

[21] Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), 5–32.

[22] Cadwalladr, C. & Graham-Harrison, E. (2018). Revealed: 50 million Facebook profiles harvested for Cambridge Analytica in major data breach. *The Guardian*, 17.

[23] Campbell, M. & Bauman, S. (2017). *Reducing Cyberbullying in Schools: International Evidence-Based Best Practices*. Academic Press.

[24] Campbell, M. A. (2005). Cyber bullying: An old problem in a new guise? *Journal of Psychologists and Counsellors in Schools*, 15(1), 68–76.

[25] Cataldi, M., Di Caro, L., & Schifanella, C. (2010). Emerging topic detection on Twitter based on temporal and social terms evaluation. In *Proceedings of the tenth international workshop on multimedia data mining* (pp.4).: ACM.

[26] Cavalin, P. R. & Cavalin, P. R. (2016). Classification of Life Events on Social Media . (October).

[27] Cavalin, P. R., Moyano, L. G., & Miranda, P. P. (2015). A Multiple Classifier System for Classifying Life Events on Social Media . *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, (pp. 1332–1335).

[28] Chen, D. & Manning, C. (2014). A Fast and Accurate Dependency Parser using Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 740–750).: Association for Computational Linguistics.

[29] Cheong, F. & Cheong, C. (2011). Social Media Data Mining: A Social Network Analysis Of Tweets During The 2010-2011 Australian Floods. *PACIS*, 11, 46–46.

[30] Cook, S. A. (1971). The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing* (pp. 151–158).: ACM.

[31] Cordeiro, M. & Gama, J. (2016). *Online Social Networks Event Detection: A Survey*, (pp. 1–41). Springer International Publishing: Cham.

[32] Cortes, C. & Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3), 273–297.

[33] Culotta, A. (2010). Towards detecting influenza epidemics by analyzing Twitter messages. In *Proceedings of the first workshop on social media analytics* (pp. 115–122).: ACM.

[34] Cunningham, H. & et al. (2011). *Developing Language Processing Components with GATE Version 6 (a User Guide)*.

[35] de Kerckhove, D. & de Almeida, C. M. (2013). What is a digital persona? *Technoetic Arts*, 11(3), 277–287.

[36] Derczynski, L., Ritter, A., Clark, S., & Bontcheva, K. (2013). Twitter Part-of-Speech Tagging for All: Overcoming Sparse and Noisy Data. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*: Association for Computational Linguistics.

[37] Deshpande, M., Kuramochi, M., Wale, N., & Karypis, G. (2005). Frequent substructure-based approaches for classifying chemical compounds. *IEEE Transactions on Knowledge and Data Engineering*, 17(8), 1036–1050.

[38] Developers, N. (2010). NetworkX. *networkx. lanl. gov*.

[39] Dickinson, T., Fernandez, M., Thomas, L. A., Mulholland, P., Briggs, P., & Alani, H. (2015a). Automatic Identification of Personal Life Events in Twitter. In *Proceedings of the ACM Web Science Conference*, WebSci '15 (pp. 37:1–37:2). New York, NY, USA: ACM.

[40] Dickinson, T., Fernandez, M., Thomas, L. A., Mulholland, P., Briggs, P., & Alani, H. (2015b). Identifying Prominent Life Events on Twitter. In *Proceedings of the 8th International Conference on Knowledge Capture*, K-CAP 2015 (pp. 4:1–4:8). New York, NY, USA: ACM.

[41] Dickinson, T., Fernandez, M., Thomas, L. A., Mulholland, P., Briggs, P., & Alani, H. (2016). Identifying Important Life Events from Twitter Using Semantic and Syntactic Patterns. In *WWW/Internet Conference proceedings 2016* (pp. 143–150).: IADIS Press.

[42] Eschweiler, S. (2016). *Angular 2*. Leanpub.

[43] Eugenio, B. D., Green, N., & Subba, R. (2013). Detecting Life Events in Feeds from Twitter . In *2013 IEEE Seventh International Conference on Semantic Computing* (pp. 274–277).

[44] Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., & Lin, C.-J. (2008). LIBLINEAR: A library for large linear classification. *Journal of machine learning research*, 9(Aug), 1871–1874.

[45] Finlayson, M. (2014). Java libraries for accessing the princeton wordnet: Comparison and evaluation. In *Proceedings of the Seventh Global Wordnet Conference* (pp. 78–85).

[46] Fournier-Viger, P. & Tseng, V. S. (2012). Mining top-K Non-redundant Association Rules. In *Proceedings of the 20th International Conference on Foundations of Intelligent Systems*, ISMIS'12 (pp. 31–40). Berlin, Heidelberg: Springer-Verlag.

[47] Green, A., Junghanns, M., Kiessling, M., Lindaaker, T., Plantikow, S., & Selmer, P. (2018). openCypher: New Directions in Property Graph Querying.

[48] Gunning, R. (1969). The Fog Index After Twenty Years . *Journal of Business Communication*, 6(2), 3–13.

[49] Hall, M. A. (1998). *Correlation-based Feature Subset Selection for Machine Learning*. PhD thesis, University of Waikato, Hamilton, New Zealand.

[50] Himsolt, M. (1997). GML: A portable graph file format. *HTML page under http://www. fmi. uni-passau. de/graphlet/gml/gml-tr. html, Universit ä t Passau*.

[51] Huan, J., Wang, W., Bandyopadhyay, D., Snoeyink, J., Prins, J., & Tropsha, A. (2004). Mining protein family specific residue packing patterns from protein structure graphs. In *Proceedings of the eighth annual international conference on Research in computational molecular biology* (pp. 308–315).: ACM.

[52] Inc, D. Docker .

[53] Inc, N. Neo4j . https://neo4j.com/.

[54] Janssen, S. M. J. & Rubin, D. C. (2011). Age Effects in Cultural Life Scripts . *Applied Cognitive Psychology*, 25(2), 291–298.

[55] Keogh, E. & Mueen, A. (2017). Curse of dimensionality. In *Encyclopedia of Machine Learning and Data Mining* (pp. 314–315). Springer.

[56] Kong, L., Schneider, N., Swayamdipta, S., Bhatia, A., Dyer, C., & Smith, N. A. (2014). A dependency parser for Tweets. In *In EMNLP*.

[57] Kotthoff, L., Thornton, C., Hoos, H. H., Hutter, F., & Leyton-Brown, K. (2017). Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA. *The Journal of Machine Learning Research*, 18(1), 826–830.

[58] Kriege, N. M. & Morris, C. (2017). Recent Advances in Kernel-Based Graph Classification. In Y. Altun, K. Das, T. Mielikäinen, D. Malerba, J. Stefanowski, J. Read, M. Žitnik, M. Ceci, & S. Džeroski (Eds.), *Machine Learning and Knowledge Discovery in Databases* (pp. 388–392). Cham: Springer International Publishing.

[59] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

[60] Kuramochi, M. & Karypis, G. (2001). Frequent subgraph discovery. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE international conference on* (pp. 313–320).: IEEE.

[61] Li, J., Ritter, A., Cardie, C., & Hovy, E. (2014). Major Life Event Extraction from Twitter based on Congratulations/Condolences Speech Acts . In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1997–2007).: Association for Computational Linguistics.

[62] Li, Y. H. & Jain, A. K. (1998). Classification of text documents. *The Computer Journal*, 41(8), 537–546.

[63] Madani, A. & Boussaid, O. (2014). What ' s Happening : A Survey of Tweets Event Detection . *INNOV 2014 : The Third International Conference on Communications, Computation, Networks and Technologies*, (c), 16–22.

[64] Mail, R. (2016). Customer Data Research Report 2016. https://www.royalmail.com/corporate/marketing-data/trends-innovation/industry-research/life-event-report.

[65] Makice, K. (2009). *Twitter API: Up and Running Learn How to Build Applications with the Twitter API*. O'Reilly Media, Inc., 1st edition.

[66] Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., & McClosky, D. (2014). The Stanford CoreNLP Natural Language Processing Toolkit . In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations* (pp. 55–60).: Association for Computational Linguistics.

[67] Massoudi, K., Tsagkias, M., de Rijke, M., & Weerkamp, W. (2011). Incorporating Query Expansion and Quality Indicators in Searching Microblog Posts . (pp. 362–367).

[68] Mathioudakis, M. & Koudas, N. (2010). Twittermonitor: trend detection over the Twitter stream. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data* (pp. 1155–1158).: ACM.

[69] Mathur, A., Lee, E., & Moschis, G. P. (2006). Life-changing events and marketing opportunities. *Journal of Targeting, Measurement and Analysis for Marketing*, 14(2), 115–128.

[70] Mathur, A., Moschis, G. P., & Lee, E. (2008). A longitudinal study of the effects of life status changes on changes in consumer preferences. *Journal of the Academy of Marketing Science*, 36(2), 234–246.

[71] Maynard, D. & Greenwood, M. A. (2014). Who cares about Sarcastic Tweets? Investigating the Impact of Sarcasm on Sentiment Analysis. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014, Reykjavik, Iceland, May 26-31, 2014.* (pp. 4238–4243).

[72] McCallum, A., Nigam, K., et al. (1998). A comparison of event models for naive Bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752 (pp. 41–48).: Citeseer.

[73] Metsis, V., Androutsopoulos, I., & Paliouras, G. (2006). Spam filtering with Naive Bayes-which Naive Bayes? In *CEAS*, volume 17 (pp. 28–69).: Mountain View, CA.

[74] Metzler, D., Cai, C., & Hovy, E. (2012). Structured event retrieval over microblog archives . *Proceedings of the 2012 Conference of the North ...*, (pp. 646–655).

[75] Miller, G. & Fellbaum, C. (1998). Wordnet: An electronic lexical database.

[76] Murphy, K. P. (2006). Naive Bayes classifiers. *University of British Columbia*, 18.

[77] N é meth, L. (2010). Hunspell. *Dostupno na: http://hunspell. sourceforge. net/[01.10. 2013]*.

[78] Navigli, R. & Ponzetto, S. P. (2012). BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193, 217–250.

[79] Nijssen, S. & Kok, J. N. (2005). The Gaston Tool for frequent subgraph mining. *Electronic Notes in Theoretical Computer Science*, 127(1), 77–87.

[80] Nivre, J. (2005). Dependency grammar and dependency parsing. *MSI report*, 5133(1959), 1–32.

[81] Owoputi, O., O'Connor, B., Dyer, C., Gimpel, K., Schneider, N., & Smith, N. A. (2013). : Association for Computational Linguistics.

[82] Parameswaran, A., Garcia-Molina, H., & Rajaraman, A. (2010). Towards the web of concepts: Extracting concepts from large datasets. *Proceedings of the VLDB Endowment*, 3(1-2), 566–577.

[83] Pham, T., Tran, T., Dam, H., & Venkatesh, S. (2017). Graph Classification via Deep Learning with Virtual Nodes. *CoRR*, abs/1708.04357.

[84] Philippsen, M., Worlein, M., Dreweke, A., & Werth, T. (2011). Parsemis: the parallel and sequential mining suite. *Available at www2. informatik. uni-erlangen. de/EN/research/ParSeMiS*.

[85] Phuvipadawat, S. & Murata, T. (2010). Breaking News Detection and Tracking in Twitter. In *2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, volume 3 (pp. 120–123).

[86] Ponte, J. M. & Croft, W. B. (1998). A Language Modeling Approach to Information Retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98 (pp. 275–281). New York, NY, USA: ACM.

[87] Popescu, A.-M. & Pennacchiotti, M. (2010). Detecting Controversial Events from Twitter. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10 (pp. 1873–1876). New York, NY, USA: ACM.

[88] Popescu, A.-M., Pennacchiotti, M., & Paranjpe, D. (2011). Extracting events and event descriptions from Twitter. *Proceedings of the 20th international conference companion on World wide web - WWW '11*, (1), 105–106.

[89] Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1), 81–106.

[90] Quinlan, R. (1993). *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann Publishers.

[91] Ramos, J. (2003). Using TF-IDF to determine word relevance in document queries.

[92] Reutemann, P. (2014). MultiSearch .

[93] Ritter, A., Clark, S., Mausam, & Etzioni, O. (2011). Named Entity Recognition in Tweets: An Experimental Study. In *Proceedings of the Conference on Empirical*

Methods in Natural Language Processing, EMNLP '11 (pp. 1524–1534). Stroudsburg, PA, USA: Association for Computational Linguistics.

[94] Rosenstiel, T., Sonderman, J., Loker, K., Ivancin, M., & Kjarval, N. (2015). How people use Twitter in general. https://www.americanpressinstitute.org/publications/reports/survey-research/how-people-use-twitter-in-general/.

[95] Rowe, M. & Alani, H. (2014). Mining and Comparing Engagement Dynamics Across Multiple Social Media Platforms. In *Proceedings of the 2014 ACM Conference on Web Science*, WebSci '14 (pp. 229–238). New York, NY, USA: ACM.

[96] Russell, S. J. & Norvig, P. (2016). *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,.

[97] S é rasset, G. (2015). DBnary: Wiktionary as a Lemon-based multilingual lexical resource in RDF. *Semantic Web*, 6(4), 355–361.

[98] Saif, H., Dickinson, T., Kastler, L., Fernandez, M., & Alani, H. (2017). A Semantic Graph-Based Approach for Radicalisation Detection on Social Media. In *ESWC 2017: The Semantic Web - Proceedings, Part I*, volume 10249 of *Lecture Notes in Computer Science* (pp. 571–587).

[99] Sail, R. (2016). How to Use Facebook Ads to Reach Couples as a Wedding Photographer.

[100] Sakaki, T., Okazaki, M., & Matsuo, Y. (2010). Earthquake shakes Twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web* (pp. 851–860).: ACM.

[101] Sankaranarayanan, J., Samet, H., Teitler, B. E., Lieberman, M. D., & Sperling, J. (2009). TwitterStand: News in Tweets. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '09 (pp. 42–51). New York, NY, USA: ACM.

[102] Scuse, D. & Reutemann, P. (2007). Weka experimenter tutorial for version 3-5-5. *University of Waikato*.

[103] Segaran, T., Evans, C., & Taylor, J. (2009). *Programming the Semantic Web: Build Flexible Applications with Graph Data. "* O'Reilly Media, Inc.".

[104] Sekine, S. & Ranchhod, E. (2009). *Named Entities: Recognition, Classification, and Use*. Benjamins current topics. John Benjamins Publishing Company.

[105] Sheldon, P. & Bryant, K. (2016). Instagram: Motives for its use and relationship to narcissism and contextual age. *Computers in Human Behavior*, 58, 89 – 97.

[106] Silva, Y. N., Hall, D. L., & Rich, C. (2018). BullyBlocker: toward an interdisciplinary approach to identify cyberbullying. *Social Network Analysis and Mining*, 8(1), 18.

[107] Singh, P., Lin, T., Mueller, E. T., Lim, G., Perkins, T., & Zhu, W. L. (2002). Open Mind Common Sense: Knowledge Acquisition from the General Public. In *On the Move to Meaningful Internet Systems, 2002 - DOA/CoopIS/ODBASE 2002 Confed-*

erated International Conferences DOA, CoopIS and ODBASE 2002 (pp. 1223–1237). Berlin, Heidelberg: Springer-Verlag.

[108] Speer, R. & Havasi, C. (2012). Representing General Relational Knowledge in ConceptNet 5. In *LREC* (pp. 3679–3686).

[109] Stone, J. V. (2013). *Bayes' Rule: A Tutorial Introduction to Bayesian Analysis*. Sebtel Press.

[110] Tarjan, R. (1972). Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2), 146–160.

[111] Thelwall, M., Buckley, K., Paltoglou, G., Cai, D., & Kappas, A. (2010). Sentiment strength detection in short informal text. *Journal of the Association for Information Science and Technology*, 61(12), 2544–2558.

[112] Thornton, C., Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2013). Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 847–855).: ACM.

[113] Tixier, A. J. . P., Nikolentzos, G., Meladianos, P., & Vazirgiannis, M. (2017). Classifying Graphs as Images with Convolutional Neural Networks. *CoRR*, abs/1708.02218.

[114] Tulving, E. et al. (1972). Episodic and semantic memory. *Organization of memory*, 1, 381–403.

[115] Twitter (2014). Building a Tweet Index.

[116] Van Rijsbergen, C. J., Robertson, S. E., & Porter, M. F. (1980). *New models in probabilistic information retrieval* . British Library Research and Development Department London.

[117] Veronis, J. (1998). Computational linguistics: special issue on word sense disambiguation.

[118] Vrande č i ć, D. & Kr ö tzsch, M. (2014). Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10), 78–85.

[119] Wang, J., Iannotti, R. J., & Nansel, T. R. (2009). School bullying among adolescents in the United States: Physical, verbal, relational, and cyber. *Journal of Adolescent health*, 45(4), 368–375.

[120] Weerkamp, W. & Rijke, M. (2008). Credibility Improves Topical Blog Post Retrieval. In *Proceedings of ACL-08: HLT* (pp. 923–931).: Association for Computational Linguistics.

[121] Werth, T., Dreweke, A., W ö rlein, M., Fischer, I., & Philippsen, M. (2008). DAGMA: mining directed acyclic graphs. In *IADIS European Conference on Data Mining* (pp. 11–18).

[122] Williams, H. L., Conway, M. A., & Cohen, G. (2008). Autobiographical memory. *Memory in the real world*, 3, 21–90.

[123] Witten, I. H., Frank, E., & Hall, M. A. (2011). *Data Mining: Practical Machine Learning Tools and Techniques*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 3rd edition.

[124] Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

[125] Xin, D., Han, J., Yan, X., & Cheng, H. (2005). Mining Compressed Frequent-pattern Sets. In *Proceedings of the 31st International Conference on Very Large Data Bases*, VLDB '05 (pp. 709–720).: VLDB Endowment.

[126] Yan, X. & Han, J. (2002). gSpan: Graph-based substructure pattern mining. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on* (pp. 721–724).: IEEE.

[127] Yan, X. & Han, J. (2003). CloseGraph: mining closed frequent graph patterns. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 286–295).: ACM.

[128] Zhang, H. (2004). The Optimality of Naïve Bayes . In *In FLAIRS2004 conference*.

[129] Zhu, N. Q. (2013). *Data visualization with D3. js cookbook*. Packt Publishing Ltd.